ABC INFORMATIQUE

N° 70

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

Photo : l'œil et la puce

Jeu de main avec Oric

L'enfant et le micro

Télécom : 01000001LLO !

EDITIONS ATLAS

M 6062-70-12,50 F 90 FB - 3,80 FS - $ 1.95

Page manquante
(publicité et colophon)

01000001LLO!

La communication entre ordinateurs signifie que des données sont transmises des uns aux autres. Comment s'établissent ces transferts qui passent le plus souvent par les lignes du téléphone?

La Voix de son Maître
Les modems (*mo*dulateurs/*dém*odulateurs) ont eu d'abord pour fonction de connecter un gros système à tous ses terminaux. Ils traduisent (« modulent ») des données électroniques, de type numérique, en signaux audio qui peuvent être envoyés via les câbles téléphoniques.
Ces signaux sont à leur tour démodulés à la réception et retrouvent leur forme primitive.
Les coupleurs acoustiques travaillent par raccord direct avec le combiné téléphonique, tandis que les modems *câblés* sont habituellement reliés directement à la ligne elle-même, généralement par l'intermédiaire d'une prise d'extension.

Il vous suffit de raccorder votre ordinateur à un modem pour pouvoir communiquer par téléphone avec d'autres micros. Vous pourrez, de cette façon, vous livrer à de multiples activités : envoyer des lettres, aussitôt reçues qu'expédiées; échanger des programmes avec des correspondants lointains; transmettre des messages publics avec d'autres usagers, ou être relié à un gros système. Examinons donc les principes qui régissent la transmission de données entre ordinateurs.

La technologie de la communication s'est d'abord développée autour des très gros systèmes. A l'origine, ceux-ci étaient placés dans une pièce spéciale, pourvue d'un système d'air conditionné, et reliés par des connexions à des terminaux dispersés dans tout le bâtiment. Et encore ces terminaux se réduisaient-ils à un écran et un clavier reliés par un câble série. L'utilisateur n'avait qu'à s'asseoir pour communiquer avec le système, à l'autre bout de l'immeuble. Tout cela fonctionnait très bien sur de courtes distances — disons quelques centaines de mètres — mais, au-delà, la détérioration du signal se révélait trop importante, même si le coût du câble ne posait pas de problème. C'est bien pourquoi les modems furent mis au point.

Un modem (terme qui vient de *mo*dulateur/*dém*odulateur) est un appareil qui permet la transmission de données à travers une simple ligne téléphonique. Les signaux électriques émis par l'ordinateur émetteur sont convertis en signaux audio, dont la fréquence et le volume sont adaptés à la diffusion sur le réseau du téléphone.

C'est ce qu'on appelle la *modulation*. Le modem de l'ordinateur récepteur convertit de nouveau ces signaux, qui reprennent leur forme originelle : c'est la *démodulation*. Un signal

constant, la *tonalité porteuse,* sert de référence pendant que les données voyagent par onde modulée. L'avantage du procédé est qu'il permet d'avoir accès à des ordinateurs lointains comme s'ils étaient directement reliés au terminal.

Celui-ci se compose d'un clavier et d'un écran, et n'est généralement pas pourvu d'un système de traitement de données (on dit alors qu'il est « non intelligent »). Mais un micro-ordinateur peut très bien en tenir lieu, pour peu qu'il fasse usage d'un logiciel approprié. De ce fait même, il peut être considéré comme un terminal « intelligent ».

Les différents types de modems

Il existe deux genres de modems : *acoustique* et *câblé*. Les premiers — qu'on appelle souvent « coupleurs acoustiques » — disposent de deux coupoles de caoutchouc sur lesquelles on pose le récepteur du téléphone. Les signaux audio sont alors transmis comme une conversation téléphonique ordinaire.

Ce sont les plus simples d'emploi, et, lorsqu'ils fonctionnent sur piles, ils peuvent au besoin être employés avec des ordinateurs portables à partir de cabines téléphoniques! Mais ils ont le gros défaut d'être très sensibles aux bruits environnants, et ne sont donc pas toujours fiables.

Les modems câblés (on dit aussi *à connexion directe*) se raccordent sur une prise jack standard, et le téléphone lui-même est directement connecté au modem. Ils sont plus sûrs et offrent généralement plus de possibilités que les modèles acoustiques. Tout cela pose des problèmes techniques et surtout administratifs.

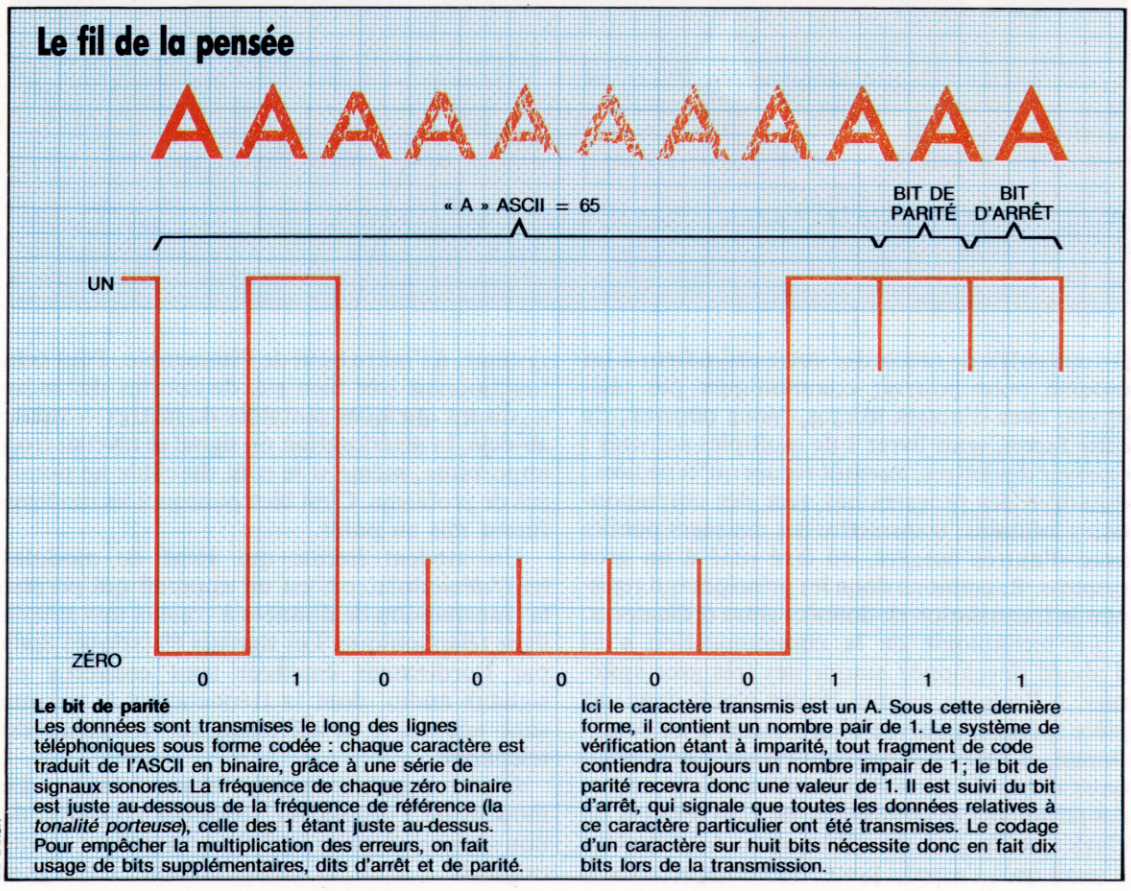
C'est ainsi que, dans la plupart des pays disposant d'une industrie nationale des télécommunications, les administrations se montrent particulièrement sourcilleuses avant de donner leur agrément pour la mise en vente de modems. Il s'agit, la plupart du temps, pour l'organisme chargé de faire respecter la loi, de voir que les tensions mises en jeu sont correctes et que les interruptions de communication laissent la ligne libre. Il est donc illégal de posséder un modem non approuvé par l'administration, mais les usagers ne semblent pas se gêner pour passer outre.

Certains types de modems offrent même l'appel et la réponse automatiques. Dans le premier cas, l'appareil accepte un numéro déterminé par l'ordinateur et l'appelle aussitôt : l'utilisateur tape un nom, le logiciel de communication recherche le numéro correspondant dans une base de données et lance le modem.

Celui-ci peut aussi répondre à un appel : si c'est un autre ordinateur, il passe le contrôle au microprocesseur, et il se contente de raccrocher si c'est un correspondant.

Le *débit en bauds* mesure la rapidité de transmission des données entre deux appareils. On le définit généralement comme étant égal au nombre de bits transmis par seconde, bien que, en pratique, ce ne soit pas tout à fait exact, pour des raisons que nous verrons plus loin.

Les deux débits les plus courants sont respectivement de 300 et de 1 200/75 (ces deux derniers chiffres signifiant que les données sont



Le bit de parité
Les données sont transmises le long des lignes téléphoniques sous forme codée : chaque caractère est traduit de l'ASCII en binaire, grâce à une série de signaux sonores. La fréquence de chaque zéro binaire est juste au-dessous de la fréquence de référence (la *tonalité porteuse*), celle des 1 étant juste au-dessus. Pour empêcher la multiplication des erreurs, on fait usage de bits supplémentaires, dits d'arrêt et de parité. Ici le caractère transmis est un A. Sous cette dernière forme, il contient un nombre pair de 1. Le système de vérification étant à imparité, tout fragment de code contiendra toujours un nombre impair de 1; le bit de parité recevra donc une valeur de 1. Il est suivi du bit d'arrêt, qui signale que toutes les données relatives à ce caractère particulier ont été transmises. Le codage d'un caractère sur huit bits nécessite donc en fait dix bits lors de la transmission.

Liz Dixon

reçues à 1 200 bauds, et transmises à 75). Ce sont d'ailleurs ceux qu'emploient presque tous les micro-ordinateurs quand ils sauvegardent des programmes sur cassette.

Un débit de 300 bauds est utile aux systèmes qui doivent transmettre dans les deux sens des quantités de données à peu près égales. C'est le cas des principaux réseaux de communication et des ensembles destinés au courrier électronique.

Mais, avec un Minitel sur le système Télétel en France, vous n'enverrez des informations qu'à la vitesse de 75 bauds actuellement. C'est encore du tourisme !

Le gros problème est que ces chiffres n'ont rien d'une norme rigoureuse. Les réseaux téléphoniques, qu'ils soient anglais, américains ou français, sont très différents, et font donc usage de fréquences qui ne sont pas les mêmes.

Des pays si proches, sur bien des points de vue, comme les États-Unis et la Grande-Bretagne ne roulent pas du même côté. Outre-Manche, le standard est le CCITT alors qu'outre-Atlantique, il s'agit du système Bell. Toutefois, si on reste sur des standards différents de part et d'autre de l'Atlantique, l'Europe a adopté la norme C.E.F.T. pour unifier Prestel (surtout en Grande-Bretagne) et Antiope (en France).

Tout sur les bits

Le réseau téléphonique ne peut employer, de façon fiable, qu'un nombre restreint de fréquences : toutes les données seront donc transmises sous forme binaire. Chaque caractère est d'abord traduit par son équivalent ASCII, lequel est à son tour converti en binaire. Un A devient d'abord un 65 (ASCII), puis un 01000001 (binaire). Les 1 de ce dernier nombre sont représentés par une fréquence (un peu supérieure à la *tonalité porteuse,* et donc appelée fréquence « haute »), et les zéros par une autre (juste au-dessous, et qui est donc la fréquence « basse »).

L'ordinateur récepteur (l'*hôte*) doit bien sûr savoir quand un caractère commence et quand il prend fin : il aura donc besoin de bits de début et de fin. Ce sont de simples fréquences prédéfinies : la réception d'un bit de fin amène par exemple le décodage des bits qui l'ont précédé. La méthode la plus commune consiste à émettre huit bits de données suivis de un bit de fin. On peut aussi envoyer sept bits, puis deux bits de fin, ou employer d'autres procédés de codage, mais le principe de base reste toujours le même.

Le téléphone n'est pas exactement un système de communication très sûr. Nous aurons, par conséquent, besoin de certaines procédures de vérification, de façon à être bien certains d'avoir reçu le bon caractère. C'est ce à quoi sert le *contrôle de parité,* qui consiste à compter le nombre de bits de fréquence haute, puis à leur ajouter un bit de parité supplémentaire, de sorte que le total soit impair (systèmes à *imparité*) ou pair (systèmes à *parité*). Revenons à la lettre A ; dans 01000001 il y a deux bits de fréquence haute (donc un nombre pair). En cas d'imparité, le bit de parité devrait lui aussi être de fréquence haute. En ce qui concerne la lettre C, en revanche, sa traduction binaire est 01000011, et le bit de parité devrait donc posséder une fréquence basse.

La vérification de parité reste assez simple dans son principe : elle permettra de détecter une erreur sur un bit isolé, mais ne pourra reconnaître deux erreurs dans le même caractère, puisque la parité sera correcte. Mais elle est amplement suffisante lorsqu'on n'a pas besoin d'une précision de cent pour cent.

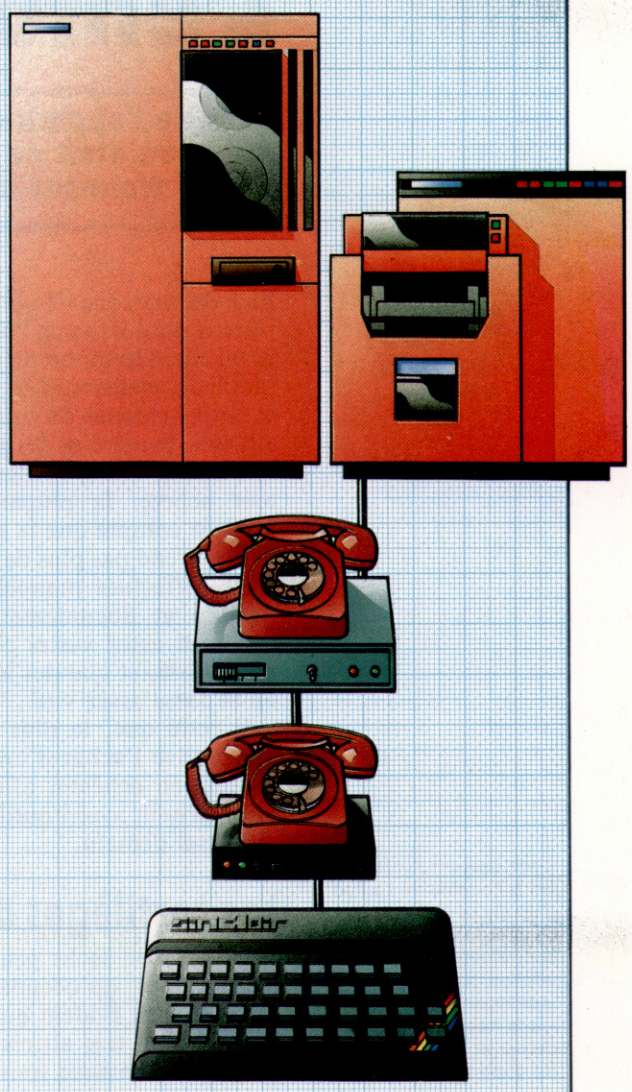
Nous faisions remarquer plus haut qu'il n'est pas tout à fait correct de définir le débit en bauds comme étant égal au nombre de bits transmis par seconde. En effet, ils contiennent également les bits de départ, d'arrêt et de parité.

Sur dix bits véhiculés par un système à parité, huit seulement contiennent des données. Le nombre réel de bits transmis par seconde est donc égal à 80 % du chiffre de 300 bauds, par exemple, soit 240 bauds.

Nous retiendrons ces principes de base de la communication des données entre modems pour, ultérieurement, nous arrêter sur plus de détails : transmission en duplex, protocoles de terminal, etc.

Le passage de l'information

Le débit en bauds (le terme a été créé en hommage à Émile Baudot, un Français pionnier de la transmission des données) mesure la vitesse à laquelle circule l'information entre deux systèmes. S'il est égal à 1, cela signifie par exemple qu'elle est transmise à raison d'un bit par seconde. En règle générale, les modems ont un débit en bauds de 300, lorsqu'il s'agit d'échanger de part et d'autre des données de masse à peu près égales, et de 1 200 lorsque le système est plus ou moins à sens unique. Ce sont là des chiffres assez semblables à ceux constatés lors de la sauvegarde ou du chargement de programmes à partir d'une cassette.
(Cl. Kevin Jones.)



L'enfant et le « micro »

Lors de l'apparition des premiers micro-ordinateurs, la grande question était de savoir ce que les individus pourraient en faire. Est-ce vraiment très différent aujourd'hui?

Bien des gens acquièrent un micro sans savoir au juste à quoi ils vont l'utiliser. Bien sûr, ils ont lu ou entendu qu'on peut traiter sa comptabilité familiale, mettre son carnet d'adresses, ses livres et autres recettes de cuisine sur ordinateur. Par ailleurs, le marché des jeux vidéo s'élargit chaque année. Mais, finalement, une fois l'ordinateur acheté, la plupart des propriétaires se rendent compte que la gestion du budget familial ne prend pas plus de temps, sauf exceptions, à être effectuée manuellement, que les bibliothèques et documentations personnelles sont rarement assez vastes pour justifier une informatisation, et que des jeux, tous comptes faits, on se lasse assez vite...

Il reste que, le plus souvent, c'est l'enfant qui motive l'achat d'un micro-ordinateur. Les parents en achètent afin que leurs rejetons se familiarisent avec l'outil de production qui sera universel demain.

Les sortilèges de l'ordinateur

Bien que l'on sache peu de chose sur la nature de la place qu'occupe l'ordinateur dans la famille, il est évident que les enfants et les adolescents sont très attirés par lui, alors que l'on perçoit encore assez souvent une certaine phobie de l'informatique de la part des adultes — phobie datant sans doute du début de l'utilisation des ordinateurs dans l'administration : si vos impôts vous sont réclamés à plusieurs reprises, si votre facture de téléphone est multipliée par cent, c'est la faute à l'ordinateur!

Il ne faut pas que cette phobie des parents devienne pour l'enfant un obstacle à la compréhension du monde de l'informatique. Toutefois, la plupart du temps, la fascination de l'écran suffit à lever l'inhibition chez l'enfant. Ce dernier fait très vite siennes ces petites machines, dont la taille se réduit progressivement comme pour se mettre mieux à sa portée. A la fascination de l'écran s'ajoute le plaisir de pianoter sur un clavier, plaisir que les enfants peuvent déjà connaître avec les petites calculatrices, sans même savoir calculer eux-mêmes. Mais à la différence du poste de télévision qui, dès qu'il s'allume, les fait interrompre leurs jeux pour assister, passifs, à une succession d'images et de sons préprogrammés, destinés à des millions de spectateurs, l'écran d'ordinateur reste vide et muet s'il n'est pas sollicité; c'est l'enfant qui va à lui, et non le contraire.

Et l'enfant va effectivement vers l'ordinateur, à tel point qu'il est capable d'assimiler des choses qui rebutent les adultes et qui le rebuteraient lui-même si elles n'étaient pas présentées sur un écran cathodique. Pour peu qu'on lui en donne l'occasion, il apprendra donc en s'amusant la table de multiplication ou les capitales de tous les pays, le calcul différentiel ou la conjugaison des verbes, ou bien tout simplement l'informatique.

Existe-t-il des programmes spécifiquement réservés aux enfants? La réponse à cette question a moins d'importance que l'affirmation implicite que contient la question elle-même : à savoir que la façon de se servir du matériel compte autant que le matériel lui-même. L'ordinateur peut vraisemblablement aider un enfant ayant des difficultés d'apprentissage; à l'opposé, il fournit des occasions à des enfants intelligents, voire plus doués que la moyenne, d'utiliser la partie de leur cerveau que l'enseignement scolaire habituel laisse généralement en friche. C'est à cette seconde catégorie que nous nous intéresserons surtout ici, la première relevant de l'éducation spécialisée. Selon Rachel Cohen, docteur ès sciences de l'éducation, qui dirige un programme d'apprentissage de l'ordinateur pour des enfants de cinq ans, « le potentiel cérébral de n'importe quel enfant est énorme. Si l'on veut qu'il se développe au maximum dans les premières années de

C'est souvent l'enfant qui est à l'origine de l'achat d'un micro-ordinateur dans la famille. Et contrairement aux adultes, il devient rapidement un partenaire « à la hauteur » du clavier et de la console. (Cl. Éditions Atlas.)

la vie, qui sont les années cruciales où les neurones se connectent, il faut plonger l'enfant dans un milieu terriblement riche où il sera saturé de stimulations ».

Que peut apprendre un enfant sur un ordinateur s'il ne sait encore ni lire ni écrire? Seul face à la machine, il ne peut évidemment pas en tirer grand-chose. La présence d'un adulte à ses côtés sera indispensable pour lui servir de guide sur les routes aventureuses de l'informatique. Dans un premier temps, des logiciels de jeux lui permettront de se familiariser avec le matériel, tout en forgeant ses réflexes et en l'obligeant à soutenir son attention et sa persévérance, qualités qui lui seront indispensables en programmation. Toutefois, si ce type de programmes donne à l'enfant l'occasion d'avoir une action visible sur l'écran tout en développant certaines qualités, il devient bientôt lassant et son intérêt est très réduit.

Les logiciels éducatifs offrent plus d'avantages et peuvent être présentés de façon suffisamment attrayante pour motiver les plus jeunes utilisateurs (voir encadré). Ainsi, grâce à l'ordinateur, un enfant peut acquérir très tôt des notions telles que la direction — droite, gauche —, l'espace, la vitesse, les nombres et même la mémoire avec ses limites ou sa capacité d'être rafraîchie. Il va, en outre, se sensibiliser très tôt à l'écrit comme à un code parallèle à l'oral. Ce code l'amènera même à apprendre à lire.

Il est facile à un programmeur un peu averti d'écrire pour un tout jeune enfant un programme affichant des lettres agrandies et de préférence colorées, correspondant à la touche appuyée par l'enfant (l'adulte à côté de lui prononçant la lettre, à défaut d'un synthétiseur vocal). L'enfant apprendra, grâce à cela, toutes les lettres de l'alphabet comme s'il s'agissait d'un jeu. Il en sera de même pour les chiffres. L'attraction de l'ordinateur et le plaisir de le faire marcher « comme papa ou maman » est généralement si fort qu'un enfant de trois ou quatre ans désirera très souvent revenir au programme qui lui a été présenté une première fois, de sorte que, avec un logiciel tel que le précédent, il connaîtra lettres et chiffres sans aucune difficulté, bien avant d'entrer à l'école primaire. En effet, l'ordinateur sollicite surtout la mémoire visuelle, laquelle est particulièrement développée chez les jeunes enfants. Il a été constaté que les enfants apprenant les lettres à l'aide d'un ordinateur distinguent mieux celles qui se ressemblent, comme le p et le q, le b et le d.

Si l'ordinateur exige, comme les jeux électroniques, une vivacité et une attention soutenues, il oblige aussi à une grande rigueur : lorsque l'enfant recopie mal un mot, l'ordinateur affiche un message d'erreur. Là encore, l'utilisateur se trouvera plus motivé : le message d'erreur n'a aucun caractère répressif qui s'apparenterait à un échec ou une sanction. C'est une invitation à réfléchir, à se poser des questions pour trouver ce qui n'est pas conforme. Il n'y a pas faute de la part de l'enfant, et d'ailleurs l'ordinateur ne tombera pas en panne parce que l'utilisateur a appuyé sur la mauvaise touche : il signale simplement à celui-ci qu'il ne le comprend pas. Il s'établit une sorte de dialogue par écrit de l'enfant avec la machine, qui implique une égalité entre adultes et enfants : devant l'ordinateur, ce n'est plus le savoir qui s'exprime, mais l'intelligence au travail. L'utilisation de l'ordinateur amène l'enfant à structurer son raisonnement, à développer son esprit de synthèse et d'analyse. Cette pratique a aussi le mérite de montrer que l'ordinateur n'est pas tout-puissant et n'a aucun caractère magique. Il faut faire découvrir à l'enfant que la machine n'est pas intelligente et que c'est l'homme (ou l'enfant) qui lui commande ce qu'elle a à faire. L'enfant découvrira aussi que, si la machine lui est supérieure du point de vue de la régularité et de la rapidité dans les actes répétitifs, il la surpasse par contre dans la création.

Initiation d'un enfant à la programmation

En recopiant de petits programmes très simples et en y ajoutant des variations, un enfant peut fort bien s'initier au mode de fonctionnement d'un ordinateur et aux principes de base de la programmation. Pour cela, il lui suffit de connaître les lettres de l'alphabet, ainsi que les chiffres. Il apprendra peu à peu à les repérer sur le clavier.

Voici, par exemple, un petit programme en BASIC qui met en évidence le caractère répétitif du traitement par ordinateur :

```
10 INPUT «NOM ?»;N$
20 FOR I=1 TO 10
30 PRINT «BONJOUR »;N$
40 NEXT I
50 END
```

Ce programme a pour effet de demander à l'enfant son nom. Lorsqu'il l'aura tapé au clavier, il verra s'afficher dix fois «BONJOUR», suivi de son nom. L'intérêt de ce programme est que l'enfant l'a écrit lui-même et qu'il peut le modifier, soit en changeant le nom, soit en faisant varier le nombre d'itérations à la ligne 20, etc.

Voici un autre programme qui souligne le caractère logique et constant de la machine. Au même programme correspond toujours le même traitement; aux mêmes données correspond toujours le même résultat :

```
10 INPUT A
20 INPUT B
30 PRINT A + B
40 END
```

Ici, l'enfant doit taper deux nombres au clavier, et l'ordinateur donne le résultat de leur addition. Si les nombres varient, le résultat différera. En revanche, le traitement reste toujours le même. L'enfant peut apporter des modifications au programme en remplaçant l'addition par n'importe quelle autre opération. Ainsi est introduite la distinction entre traitement et données.

Pour approfondir cette idée, il est essentiel d'expliciter la notion d'algorithme, c'est-à-dire le cheminement logique comportant une succession de traitements, menant de l'énoncé d'un problème à sa solution. L'enfant comprendra facilement cette notion si elle est introduite à l'aide d'exemples qui lui sont familiers et concrets. En voici un, permettant de bien faire la distinction entre données et traitement :

le problème consiste à faire de la confiture de fraises;

les données sont les fraises et le sucre, et le traitement consiste en la succession des opérations suivantes : mélange des ingrédients, cuisson durant une demi-heure, mise en bocaux;

le résultat est la confiture.

Si l'une des données ou l'une des étapes du traitement manque, le résultat n'est pas atteint.

L'enfant est ensuite invité à énoncer un autre problème ou bien une variante du premier, afin de le résoudre de la même manière que le précédent.

Des exercices de ce type, tout comme la programmation, exigent une logique rigoureuse qui favorise à la fois l'esprit d'analyse, l'esprit de synthèse et le sens de l'organisation : un organigramme, schéma de l'algorithme, ressemble à un jeu de constructions où tous les niveaux s'emboîtent à merveille.

Boucles sans fin

Pour avoir la possibilité de répéter une série d'opérations autant de fois que nous le souhaitons, il nous faut introduire la notion de « boucle », très utile en programmation.

Opération
Ce symbole représente l'exécution d'une opération ou d'un ensemble d'opérations qui peuvent être des calculs arithmétiques, des changements ou substitutions de valeurs, des modifications de données ou des changements de leurs positions, etc.

Prenons comme exemple une opération arithmétique. Nous avons trois fichiers, et chacun contient une valeur numérique. Nous cherchons à visualiser la valeur de la somme de ces quantités et aussi le carré du contenu de chaque fichier.

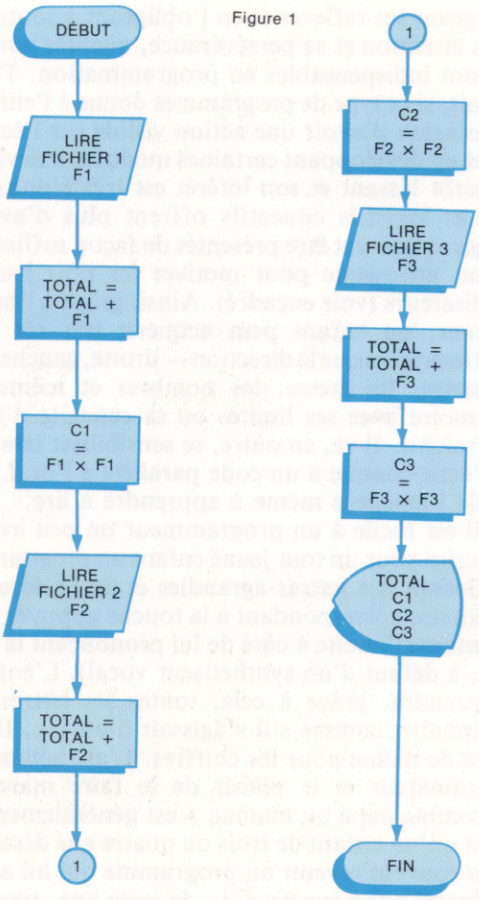
Pour cela, nous avons d'abord besoin d'une série de zones destinées à contenir les données que l'on veut visualiser. Dans l'exemple montré en figure 1, nous utilisons les zones suivantes : TOTAL pour désigner celle dans laquelle on additionne au fur et à mesure les différentes quantités, et C1, C2 et C3 pour indiquer les carrés des fichiers correspondants.

Le processus s'établit alors de la façon suivante : lire un fichier, la quantité qui y est contenue s'accumule dans la zone TOTAL ; le carré de cette quantité est ensuite calculé tout en restant dans la zone correspondante. Une fois réalisées ces opérations sur les trois fichiers, nous visualisons les zones qui contiennent les trois carrés et celui de la somme correspondant à la somme des trois fichiers.

Le déroulement de ce processus peut être considéré comme valable pour des exemples aussi simples que celui qui vient d'être décrit, où l'on compte un nombre très limité de fichiers. Mais supposons que l'on doive travailler sur un nombre de fichiers très supérieur (50 ou 100). La tâche est non seulement plus longue, mais aussi répétitive, car nous pouvons observer qu'une même partie du schéma se répète : précisément celle qui correspond à la lecture du fichier, à la somme des valeurs et au calcul du carré.

La figure 2 nous montre comment reproduire une même partie du processus. Après l'entrée de la quantité appartenant à un fichier, la somme est obtenue dans la zone TOTAL, puis la vérification du carré, et enfin la visualisation de l'ensemble des zones. Immédiatement après, la ligne de flux retourne au point initial, et le processus recommence. Dans de tels processus, il n'existe pas une fin logique, et l'on obtiendra une répétition sans fin des opérations à l'image d'un cercle fermé à l'intérieur duquel se réalise une partie du processus. Cette forme de répétition se nomme « boucle inconditionnelle ». Ce nom s'explique de la manière suivante : une fois arrivé au point où apparaît la ligne conductrice, le flux suit automatiquement sa direction, ce qui le transforme en une sorte de recyclage infini. Il ne s'arrêtera que par l'intervention de l'opérateur provoquant une rupture volontaire du cycle. Nous verrons prochainement comment établir cette rupture de cycle.

Figure 1



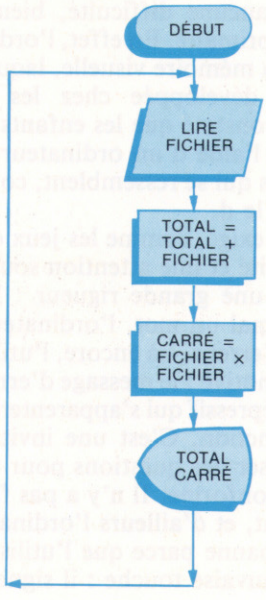
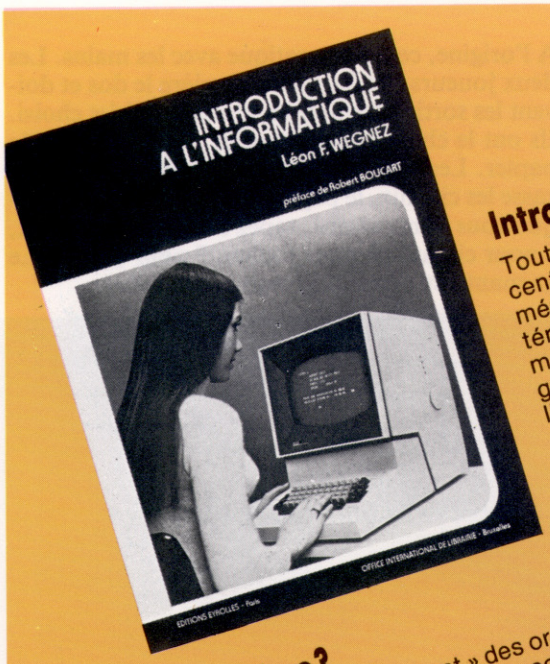


Figure 2

Livres pour débuter

Voici une sélection de livres pour vous initier à ces étranges et merveilleuses machines que sont les micro-ordinateurs. Ils vous feront entrer dans le cercle des amateurs éclairés, puis des passionnés de micro-informatique.



Introduction à l'informatique

Tout ce que contient un ordinateur : unité centrale, entrées et sorties, divers types de mémoires et supports d'information. Les caractéristiques des différents langages de programmation et exemples d'instructions. Divers usages de l'ordinateur. Derniers développements de l'informatique. Terminologie de base.

Par L.F. Wegnez.
220 pages, format 21 × 27 cm.
Prix : 118 F.
Eyrolles.

Vous avez dit Micro?

En vous enseignant le « raisonnement » des ordinateurs, cette méthode vous permettra de commencer à programmer si vous êtes débutant, ou de vous perfectionner si vous êtes déjà informaticien amateur. Vous saurez analyser un problème, en élaborer l'organigramme, réaliser le programme en BASIC et le mettre au point. Cette initiation est complétée par de nombreuses explications sur la technologie et les principes de fonctionnement des micro-ordinateurs.

Par M. Marchand.
224 pages, format 15 × 21 cm.
Prix : 92 F.
E.T.S.F.

Papa, maman, l'ordinateur et moi

« Le premier roman illustré de la micro-informatique pour tous » conte l'histoire humoristique (et réaliste) d'une famille aux prises avec un micro-ordinateur.
Il prend le lecteur par surprise, le fait rire, lui fait assimiler (à son insu) tous les secrets de la micro-informatique et éviter tous ses pièges, depuis le choix de l'ordinateur jusqu'à son utilisation ou sa programmation.

Par S. Raven.
190 pages, format 15,5 × 24 cm.
Prix : 55 F.
S.E.C.F., Éditions Radio.

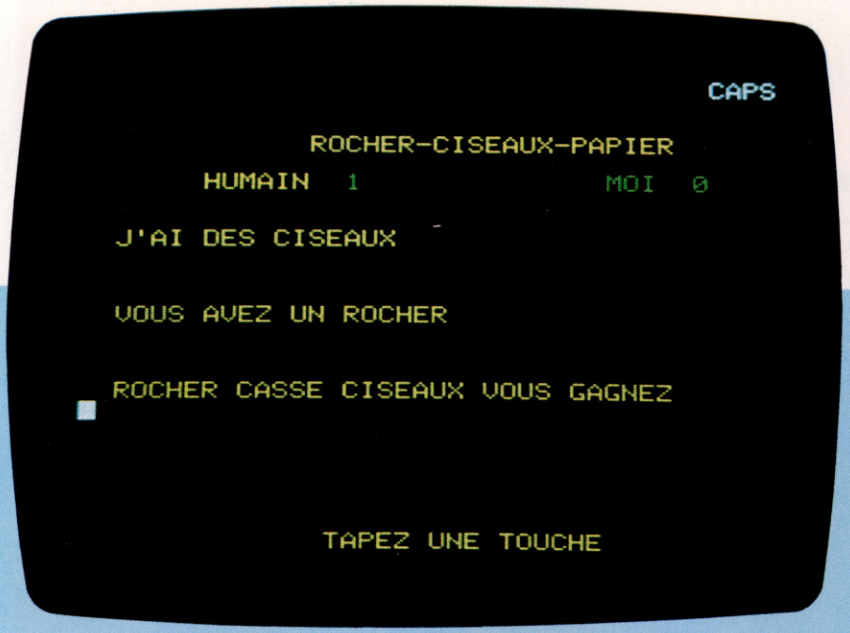
Comprendre l'informatique

Ce livre a pour but de faire comprendre en profondeur, et cependant sans lourde technicité, ce qu'est, ce que fait, comment calcule, comment s'emploie un ordinateur.

Par É. Labin.
256 pages.
Prix : 87 F.
Bordas initiation.

Main à main

Le succès de ce jeu, plus connu sous l'expression « rocher, ciseaux et papier », ne peut être ignoré par votre ordinateur. Voici une version pour l'Oric, écrite en BASIC par Peter Shaw.



A l'origine, ce jeu se pratique avec les mains. Les deux joueurs ont les mains derrière le dos et doivent les sortir ensemble en mimant l'objet choisi. Ils ont le choix entre le rocher, les ciseaux, et le papier. Les ciseaux coupent le papier, le rocher casse les ciseaux, le papier couvre le rocher. Lorsque vous aurez choisi, tapez l'initiale de l'objet, C pour ciseaux, R pour rocher, P pour papier. Le gagnant est celui qui totalise dix points.

```
10 REM ROCHER, PAPIER, CISEAUX
20 CLS'   PETER SHAW **
30 PAPER 0:INK 3
40 A=INT(RND(1)*3):PING
50 PLOT 12,2,"ROCHER-CISEAUX-PAPIER"
55 PLOT 6,4,"HUMAIN "+STR$(HS)
56 PLOT 29,4,"MOI "+STR$(OS)
60 GET A$:ZAP:FOR P=0 TO 7:INK P:WAIT 4:
NEXT P
65 CLS
70 IF A$="R" THEN V=0
80 IF A$="P" THEN V=1
90 IF A$="C" THEN V=2
100 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:
PRINT
110 PRINT"J'AI ";
120 IF A=0 THEN PRINT"UN ROCHER"
130 IF A=1 THEN PRINT"DU PAPIER"
140 IF A=2 THEN PRINT"DES CISEAUX"
150 PRINT:PRINT:PRINT
160 PRINT"VOUS AVEZ ";
170 IF V=0 THEN PRINT"UN ROCHER"
180 IF V=1 THEN PRINT"DU PAPIER"
190 IF V=2 THEN PRINT"DES CISEAUX"
200 PRINT:PRINT:PRINT:SHOOT
210 IF V=A THEN PRINT"EGALITE"
220 IF V=0 AND A=1 THEN PRINT"PAPIER COU
VRE ROCHER JE GAGNE":OS=OS+
1
230 IF V=0 AND A=2 THEN PRINT"ROCHER CAS
SE CISEAUX VOUS GAGNEZ":HS=
HS+1
240 IF V=1 AND A=0 THEN PRINT"PAPIER COU
VRE ROCHER VOUS GAGNEZ":HS=
HS+1
250 IF V=1 AND A=2 THEN PRINT"CISEAUX CO
UPENT PAPIER JE GAGNE":OS=O
S+1
260 IF V=2 AND A=0 THEN PRINT"ROCHER CAS
SE CISEAUX JE GAGNE":OS=OS+
1
270 IF V=2 AND A=1 THEN PRINT"CISEAUX CO
UPENT PAPIER VOUS GAGNEZ":H
S=HS+1
280 PLOT 13,23,"TAPEZ UNE TOUCHE"
290 EXPLODE:WAIT 30
295 IF HS=10 OR OS=10 THEN 310
300 GOTO 30
310 CLS
320 IF HS=10 THEN PING:PRINT"BRAVO VOUS
GAGNEZ"
330 IF OS=10 THEN ZAP:PRINT"JE GAGNE ENC
ORE"
340 PRINT:PRINT:PRINT
345 WAIT 20:EXPLODE
350 END
```

L'œil et la puce

Les derniers appareils photographiques exploitent la puissance des microprocesseurs pour faciliter les prises de vue. Nous passons en revue certains de ces appareils qui sont offerts sur le marché.

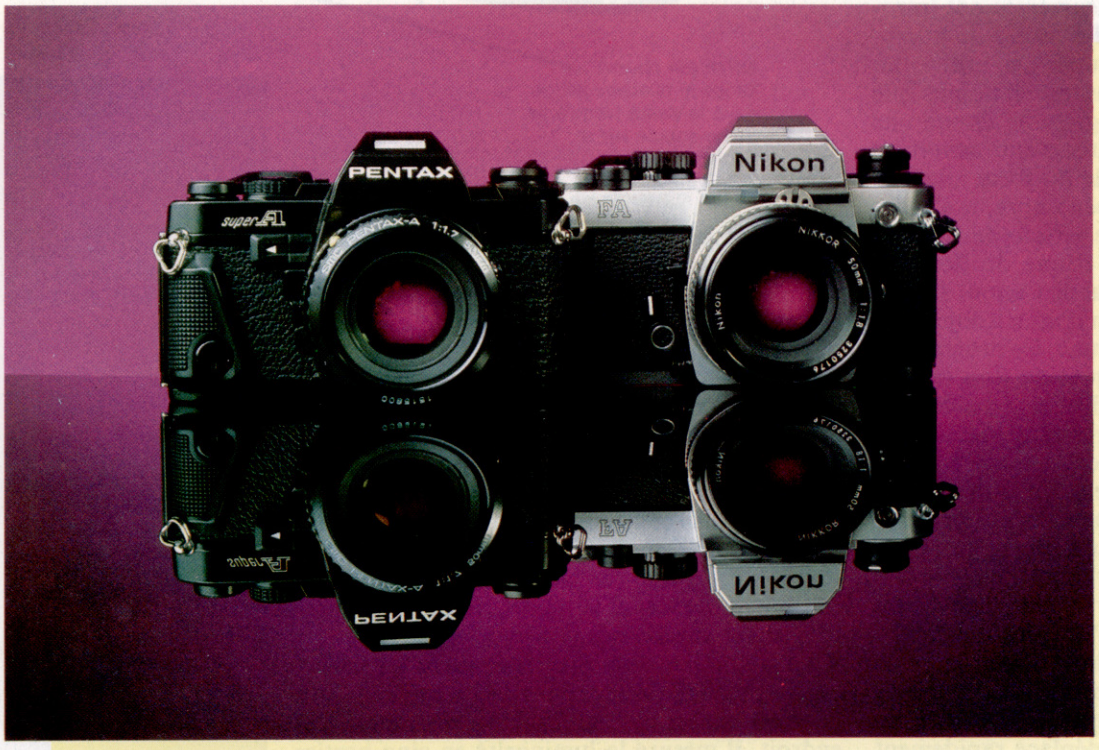
Pour toute prise de vue, la première tâche du photographe consiste à trouver la meilleure exposition. Il doit évaluer la quantité de lumière qui atteindra le film dans son appareil : trop forte, la photographie sera pâle et sans relief ; trop faible, l'image sera trop sombre. Pour obtenir une exposition correcte, il est nécessaire de trouver un juste équilibre entre l'ouverture de l'objectif et la vitesse d'obturation qui détermine la durée de l'exposition.

La quantité de lumière émise par une scène particulière doit d'abord être mesurée. Puis, en tenant compte de la sensibilité du film, il convient de calculer l'ouverture et la vitesse d'obturation. Des posemètres furent mis au point pour permettre aux photographes de mesurer avec précision la luminosité d'une scène donnée. Plus récemment, des cellules photoélectriques furent intégrées dans les appareils, ne dispensant pas le photographe de devoir toujours sélectionner une vitesse d'obturation et une ouverture correspondant à la lecture de la cellule.

Les progrès réalisés dans le domaine de l'électronique au cours des années soixante-dix permirent de définir automatiquement les réglages d'ouverture et de vitesse d'obturation à partir de la lecture de la cellule. Sans l'intervention du photographe, il est possible d'obtenir d'« excellents » résultats en dirigeant l'appareil vers un sujet et en appuyant simplement sur le bouton de l'obturateur. La manœuvre est très utile pour le débutant qui désire réaliser des photographies sans comprendre le fonctionnement de l'appareil, et pour le photographe de reportage qui travaille souvent dans des conditions difficiles.

Le Canon A1 permet de choisir parmi six priorités différentes pour faire une photo :

1. *Priorité à la vitesse :* l'utilisateur choisit une vitesse et l'appareil détermine l'ouverture correspondante.
2. *Priorité à l'ouverture :* l'utilisateur choisit une ouverture et l'appareil définit la vitesse d'obturation.
3. *Programme :* l'appareil détermine la vitesse et l'ouverture à l'aide d'un programme qui produit une combinaison optimale des deux facteurs.
4. *Flash automatique :* lorsque certains flash sont posés sur l'appareil, ce dernier sélectionne la vitesse adéquate (1/60 de seconde) et détermine



Réglages automatiques Depuis le début des années soixante-dix, des microprocesseurs commandent les fonctions de calcul de la lumière et d'ouverture d'objectif. Aujourd'hui, les appareils commandés par microprocesseurs ont un potentiel beaucoup plus grand : ils se chargent de la vitesse, de l'ouverture, des fonctions du flash, et permettent ainsi à l'utilisateur débutant de produire d'excellentes photographies, même dans des conditions difficiles de luminosité. C'est le cas du Nikon FA et du Pentax Super A qui grâce à divers programmes prennent en compte des situations variées. (Cl. Ian McKinnell.)

l'ouverture correspondante du flash. Une cellule intégrée au flash coupe l'alimentation de celui-ci si le sujet est suffisamment éclairé.

5. *Priorité à l'ouverture réelle :* ce choix est destiné à d'anciens objectifs et à certains accessoires qui travaillent toujours en ouverture réelle, c'est-à-dire à la valeur d'ouverture utilisée pour réaliser la photo. Les objectifs modernes conservent l'ouverture maximale afin d'obtenir dans le viseur une image aussi lumineuse que possible.

6. *Manuelle :* la vitesse et l'ouverture sont choisies par le photographe. Cela permet un contrôle total lorsque l'on désire créer des effets spéciaux ou lorsque la scène présente un éclairage en contre-jour.

Un affichage électronique à l'intérieur du viseur du Canon A1 indique le mode dans lequel se trouve l'appareil et les valeurs d'ouverture et de vitesse sélectionnées. L'affichage est produit au moyen de diodes électroluminescentes ; il peut donc être consulté dans l'objectif.

Malgré ses avantages, le mode programme du Canon A1 comporte aussi des lacunes. Dans certains cas, il ne choisit pas la meilleure combinaison possible. Par exemple, si l'on désire photographier au crépuscule, la caméra peut sélectionner une vitesse de 1/30 de seconde et une ouverture de 2,8. Toute photographie prise à une vitesse inférieure à 1/60 de seconde risque d'être gâchée par le moindre tremblement de la main du photographe. Ce danger est signalé par un message clignotant lorsque la vitesse est inférieure à 1/60 de seconde, mais le programme ne sélectionne pas une plus grande ouverture pour autoriser une vitesse supérieure.

Compétition

De nombreuses sociétés concurrentes ont introduit des appareils multimodes munis de microprocesseurs. Le Pentax Super A utilise un programme légèrement plus perfectionné que celui du Canon A1 ; ce programme donne une meilleure combinaison de vitesse et d'ouverture dans des conditions de faible ou forte intensité lumineuse. En présence de pénombre, le Pentax Super A sélectionnerait une vitesse de 1/60 de seconde afin de réduire le risque de flou. Le Nikon FA (l'autre grand rival de Canon) signale automatiquement l'utilisation d'un téléobjectif et fait appel à un autre programme. Ce programme est destiné à limiter les risques de flou lors de l'utilisation de téléobjectifs ; il choisit des vitesses plus rapides et des ouvertures plus grandes.

Suivant l'exemple de Nikon, Canon a utilisé trois programmes optionnels sur son dernier appareil, le Canon T70. L'un de ces programmes est destiné aux objectifs ordinaires, un autre aux téléobjectifs et un troisième concerne les objectifs grand angle. Cependant, l'appareil ne reconnaît pas automatiquement l'objectif qui lui est adapté. L'utilisateur doit donc sélectionner le programme approprié. Une bonne occasion pour devenir plus créatif. Par exemple, si vous photographiez un objet qui se déplace rapidement avec un objectif grand angle, vous pouvez sélectionner le mode téléobjectif pour obtenir des vitesses rapides afin de geler l'action.

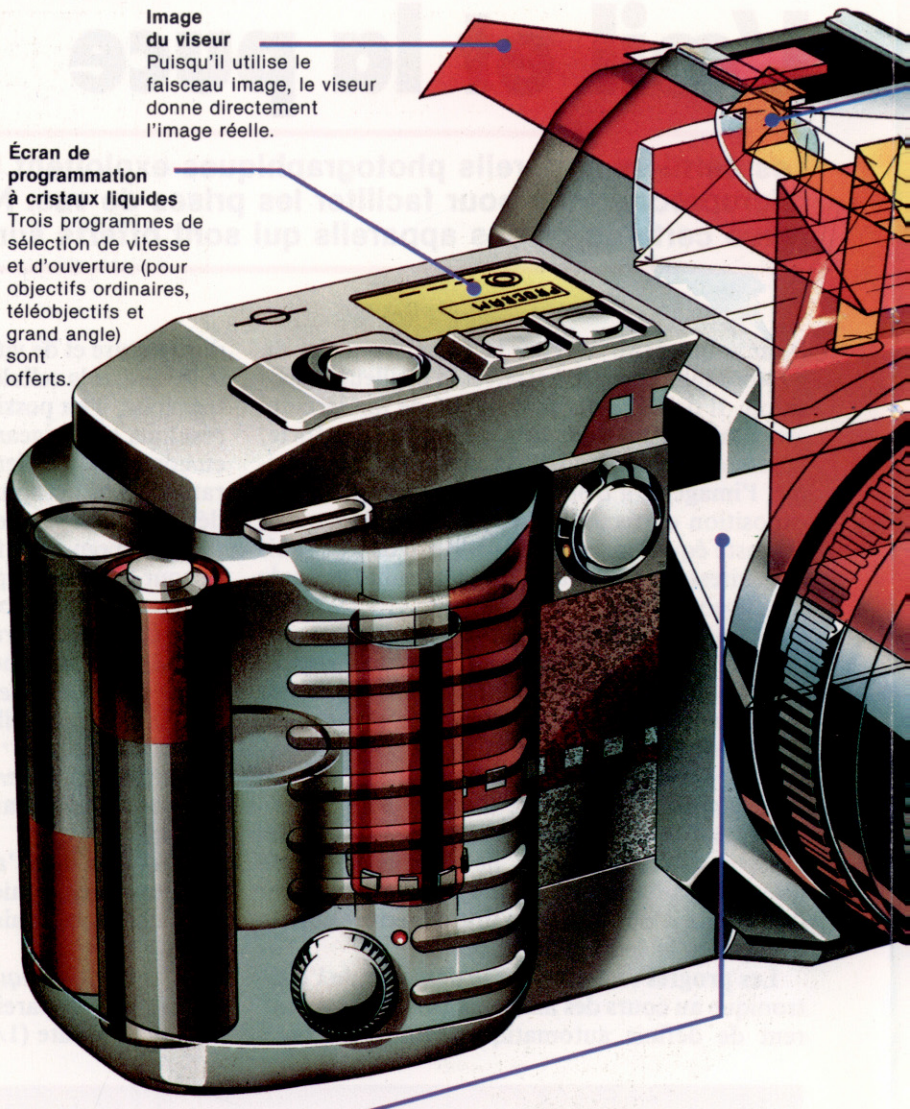


Image du viseur
Puisqu'il utilise le faisceau image, le viseur donne directement l'image réelle.

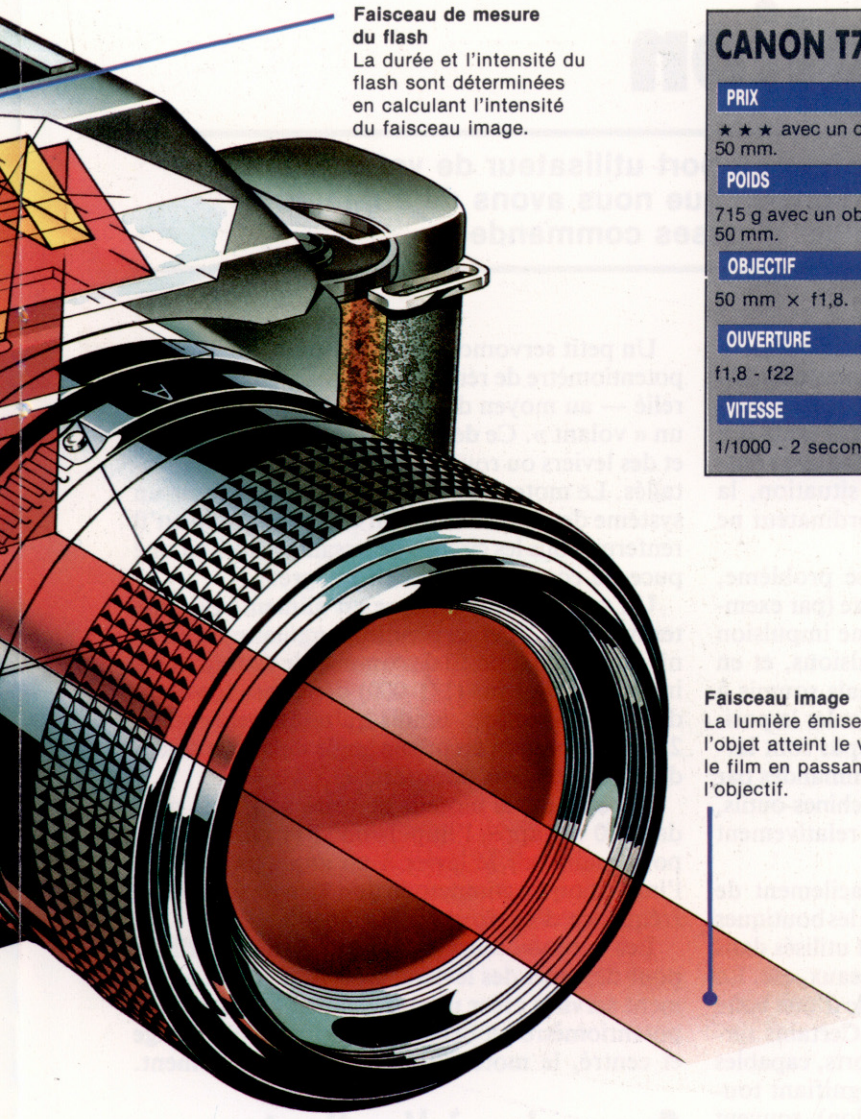
Écran de programmation à cristaux liquides
Trois programmes de sélection de vitesse et d'ouverture (pour objectifs ordinaires, téléobjectifs et grand angle) sont offerts.

Miroir sur ressort
Dirige le faisceau image dans le viseur jusqu'à ce qu'on appuie sur le déclencheur de l'obturateur.

Les appareils automatiques ont également l'inconvénient de calculer une exposition moyenne pour l'image globale ; la cellule peut donc facilement être trompée par des sujets comportant une plage étendue de luminosité. Par exemple, si une motocylette est photographiée devant un coucher de soleil, l'appareil aura tendance à calculer la lecture en fonction de la luminosité du soleil et la moto sera donc sous-exposée. Si la moto était photographiée devant un mur très sombre, l'appareil verrait le sujet beaucoup plus sombre qu'il n'est et la photographie serait probablement surexposée.

Le Nikon FA, à un prix légèrement supérieur, contourne le problème d'une façon originale. Au lieu de mesurer la luminosité du sujet à un seul endroit, il mesure la luminosité en cinq parties différentes. Le FA utilise alors un microproces-

Photographe assisté par ordinateur
Une unité centrale à huit bits construite sur commande pilote le fonctionnement du Canon T70. L'oscillateur de synchronisation à cristal produit les impulsions d'horloge et commande la durée d'exposition.
Des contacts à ressorts commandés par les circuits de la cellule règlent l'ouverture.
Un module de commande optionnel peut être ajouté et donne un intervalle d'exposition automatique (compris entre une seconde et un jour) et permet d'écrire les données de minutage directement sur le négatif.



CANON T70

PRIX

★ ★ ★ avec un objectif de 50 mm.

POIDS

715 g avec un objectif de 50 mm.

OBJECTIF

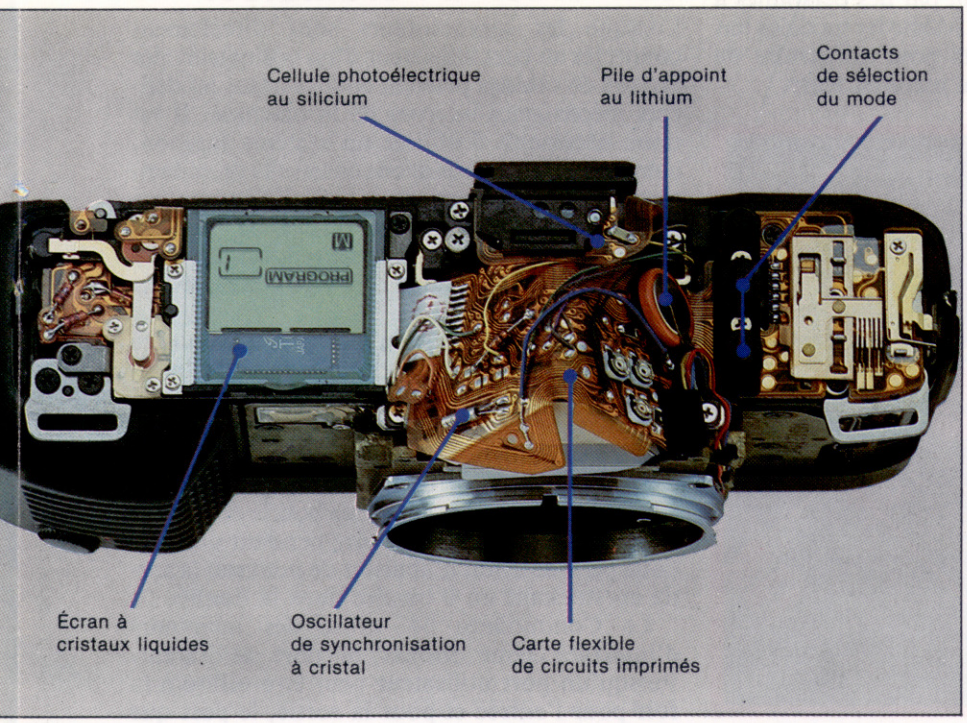
50 mm × f1,8.

OUVERTURE

f1,8 - f22

VITESSE

1/1000 - 2 secondes.



seur pour comparer les cinq lectures à diverses « scènes standard » programmées dans l'appareil. Chacune de ces scènes est obtenue après analyse de milliers de photographies.

Mais parmi les appareils actuellement offerts sur le marché, le Canon T70 est le seul qui, à un prix abordable, exploite entièrement les nouvelles capacités offertes par l'électronique. Le 170 n'a aucune commande mécanique; tous ses réglages sont effectués en appuyant sur des boutons. L'un des huit modes peut être sélectionné en appuyant sur un bouton situé dans le coin droit de l'appareil. Une information apparaît sous forme d'un affichage à cristaux liquides. D'importantes informations, comme la vitesse, l'ouverture et le mode, apparaissent également dans le viseur; le photographe peut donc regarder continuellement dans le viseur pendant les opérations de réglage.

Fonctionnement du Canon T70

Quant à la sélection de la vitesse, deux boutons permettent de l'accroître ou de la diminuer, et sa valeur est affichée sur l'écran à cristaux liquides. La vitesse du film (sa sensibilité, nombre ASA ou ISO) est réglée un peu de la même manière. Le compteur de clichés apparaît également sur l'affichage à cristaux liquides.

L'appareil est muni d'un moteur intégré qui fait avancer le film et le rebobine lorsque la pellicule est terminée. Le T70 est alimenté par deux piles ordinaires. Leur état est indiqué au moyen de trois barres sur l'affichage à cristaux liquides. Si les trois barres apparaissent, les piles sont neuves; la présence d'une seule signale que les piles doivent être remplacées. Si le déclencheur automatique de l'appareil est utilisé, le compte à rebours apparaît sur l'affichage à cristaux liquides jusqu'au déclenchement de l'obturateur.

Les microprocesseurs utilisés dans les appareils photographiques sont beaucoup moins puissants que ceux utilisés dans les ordinateurs. Celui du T70, spécialement conçu, est un huit bits; il est de type CMOS pour réduire la consommation électrique. Sa vitesse d'horloge n'est que 32 KHz; les ordinateurs fonctionnent environ 100 fois plus rapidement. Lorsqu'il n'est pas utilisé, l'appareil passe à une vitesse de 8 KHz pour économiser de l'énergie.

Le microprocesseur, contenu dans un ensemble plat à 60 broches, est doté d'une grande capacité de ROM, mais seulement de 16 octets de RAM. Parmi les quatre autres puces qui composent le microprocesseur, la plus importante est celle d'entrée/sortie. Elle commande le fonctionnement mécanique de l'appareil au moyen d'aimants et d'un moteur. Elle convertit aussi le signal électrique provenant de la puce de la cellule en un signal numérique traité par le microprocesseur.

La puissance de la micro-électronique facilite grandement la réalisation d'excellentes photographies et élimine la nécessité de connaître à fond tous les détails techniques photo. Mais il est évident que dans certains cas l'appareil choisira la mauvaise exposition ou une vitesse inadéquate.

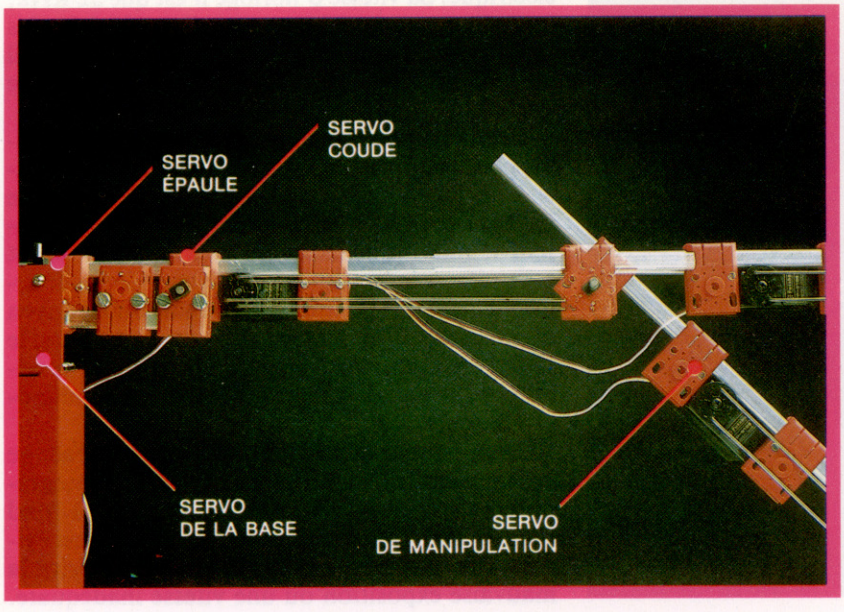
Alimentation

En connectant un servomoteur au port utilisateur de votre ordinateur et en utilisant le système tampon que nous avons déjà mis au point, il est possible d'utiliser diverses commandes.

Il existe trois types de moteurs électriques : à courant continu, pas à pas, et servomoteur. Un moteur à courant continu (C.C.) peut être facilement commandé par un ordinateur, mais il a tendance à être imprécis s'il rencontre une résistance quelconque. Dans une telle situation, la vitesse du moteur est réduite, et l'ordinateur ne peut plus le contrôler.

Un moteur pas à pas n'a pas ce problème, puisqu'il se déplace selon un angle fixe (par exemple 7,5°) chaque fois qu'il reçoit une impulsion de courant. En comptant les impulsions, et en supposant que le moteur n'est jamais soumis à une surcharge, l'ordinateur peut calculer la position du moteur. Les moteurs pas à pas sont largement utilisés dans les systèmes commandés par ordinateur — les bras-robots, les machines-outils, les trieuses... — même s'ils sont relativement coûteux.

Il est possible de se procurer facilement de petits servomoteurs numériques dans les boutiques de modélisme; ils sont fréquemment utilisés dans les modèles réduits d'avions, de bateaux, etc. La taille de ces moteurs peut aller de celle d'une boîte d'allumettes à dix fois cette taille. Certains servomoteurs sont remarquablement forts, capables de fournir un couple (expression signifiant toutes forces qui provoquent une rotation), souvent plus élevé que celui produit par des personnes à l'aide d'un gros tournevis. Même les modèles bas de gamme sont excellents pour construire de petits bras-robots et autre matériel léger.



Bras-robot
Le bras-robot Beasty est actionné par trois servomoteurs (rotations au niveau de la base, de l'épaule et du coude) avec un quatrième servomoteur servant à actionner l'extrémité du bras. Le servomoteur est parfaitement indiqué dans le cas d'un bras-robot, parce qu'il peut être bloqué après avoir été déplacé.
(Cl. Ian McKinnell.)

Un petit servomoteur numérique renferme un potentiomètre de réaction et un petit moteur C.C. relié — au moyen d'une série d'engrenages — à un « volant ». Ce dernier est situé sur le moteur, et des leviers ou roues à picots peuvent y être installés. Le moteur est aussi idéal pour réaliser un système de réaction avec un ordinateur puisqu'il renferme tous les circuits nécessaires, ainsi qu'une puce de contrôle de circuit intégré.

Un servomoteur typique est alimenté par une tension de 5 V, et sa position angulaire est définie au moyen d'un fil de commande distinct. Une impulsion de 1 ms (1/1 000) déplacera le volant dans une direction, tandis qu'une impulsion de 2 ms provoquera un même angle de déplacement dans la direction opposée.

Cependant, le moteur ne reste actif que pendant 20 ms après l'impulsion. Par conséquent, pour maintenir le levier à un angle particulier, l'impulsion de commande doit être répétée à une fréquence d'environ 50 Hz.

Les servomoteurs sont normalement utilisés pour déplacer des leviers, mais ils peuvent également servir à créer un mouvement linéaire. Si le potentiomètre est découplé du train d'engrenage et centré, le moteur tournera continuellement.

Connexion à l'ordinateur

Lorsque les servomoteurs sont directement connectés au port utilisateur d'un ordinateur, des erreurs de câblage peuvent endommager les délicats mécanismes internes de la machine. Il est donc nécessaire d'utiliser un système tampon, comme ceux que nous avons construits précédemment. Vous devez connecter le fil d'alimentation du moteur à la prise positive (rouge) de l'une des lignes du boîtier de sortie basse tension. Le fil de mise à la terre commune doit être connecté à la prise négative (noire).

Si le fil de commande du moteur a été connecté à la ligne de donnée 0 du port utilisateur de l'ordinateur, le moteur lui-même peut être commandé en envoyant les séquences d'impulsions appropriées à cette ligne 0. Une impulsion est envoyée en portant la ligne 0 à 5 V, en stockant un 1 binaire dans la donnée 0. Une boucle de compte à rebours sert alors à attendre pendant une certaine période avant d'abaisser de nouveau la sortie en stockant un 0 binaire dans la donnée 0.

Le Commodore 64 utilise des adapteurs d'interface pour former le port utilisateur. Puisqu'un port utilisateur peut être utilisé à la fois pour l'entrée et pour la sortie, le port que

nous utilisons doit d'abord être réglé au mode demandé (ici, en sortie). Cela est obtenu en écrivant des données directement en BASIC dans le registre de commande de direction.

Voyons maintenant comment envoyer des impulsions à une ligne de données du port utilisateur. Si une valeur hexadécimale de 88 (équivalent au nombre décimal 136, ou configuration binaire de 10001000) était écrite dans le port utilisateur, les tensions des broches de données seraient : 5 V, 0 V, 0 V, 0 V, 5 V, 0 V, 0 V, 0 V, respectivement. Cette configuration serait conservée jusqu'à ce qu'elle soit délibérément modifiée. Par conséquent, une simple impulsion peut être générée sur la ligne de données 0 en écrivant hex 00, hex 88, hex 00.

Pour créer une impulsion assez rapide, on doit utiliser le langage machine. Voici l'algorithme qui servira à envoyer les impulsions à un servomoteur :

1. Spécifier l'angle du levier en le stockant dans un octet (nommé ANGLE) avec une valeur comprise entre 0 et 255, à l'aide d'un programme BASIC.
2. Mettre la ligne de données 0 au niveau élevé (5 V), ce qui provoque l'impulsion.
3. Attendre 1 ms en empruntant une boucle et en décrémentant le compteur.
4. Attendre une période supplémentaire comprise entre 0 et 1 ms, de nouveau au moyen d'une boucle, mais, cette fois, la valeur de départ du compteur (par conséquent le nombre de boucles) est ANGLE.
5. Ramener la ligne de données 0 au niveau bas (0 V) pour mettre fin à l'impulsion.

Si ANGLE = 0, l'impulsion durera 1 ms; si ANGLE = 128, elle durera 1,5 ms, et le levier se déplacera à mi-parcours.

Un train d'impulsions

Une impulsion n'est néanmoins pas suffisante pour maintenir la position d'un servomoteur. Il doit recevoir un train d'impulsions, réactivant le moteur à chaque 20 ms. Il y a deux façons de produire un train d'impulsions :

- en utilisant une boucle d'attente pour marquer une pose entre les impulsions. Cela signifie que l'ordinateur ne peut rien faire d'autre pendant la durée de la boucle;
- en utilisant des « interruptions » qui permettent à l'ordinateur d'exécuter un autre programme — généralement en BASIC — presque simultanément. Ce programme de fond peut commander les mouvements des moteurs.

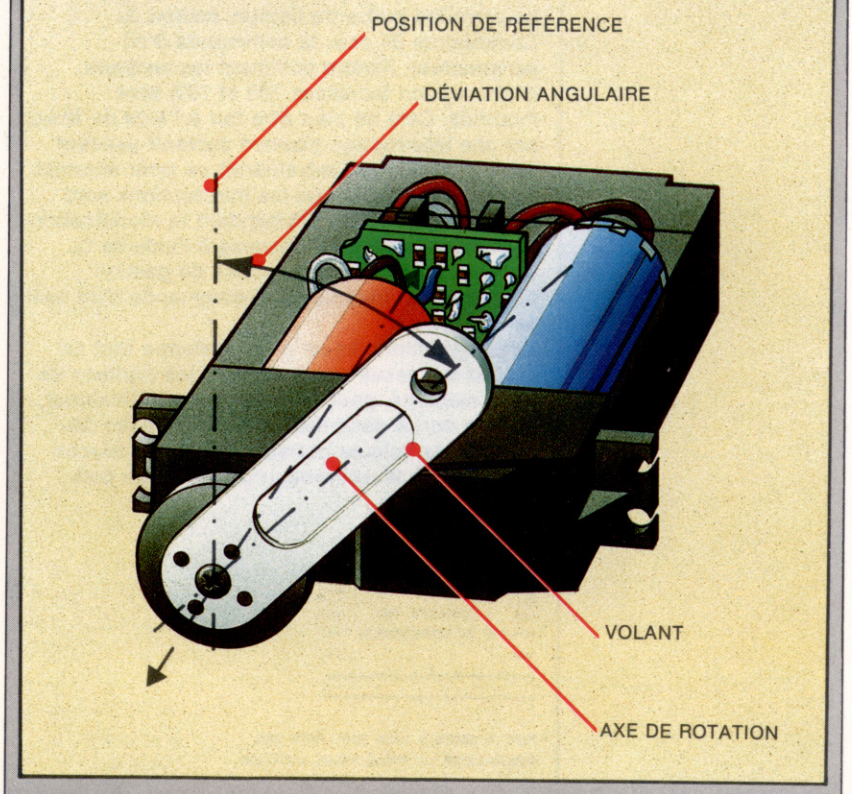
Le Commodore 64 utilise des processeurs de la série 6500 qui ont deux broches d'interruptions — NMI et IRQ. La seconde, la ligne d'interruption, sera utilisée pour vos tâches de synchronisation. Lorsqu'une impulsion apparaît sur la ligne IRQ, le processeur interrompt le traitement en cours et commence à exécuter le programme de gestion d'interruption. Une fois terminé, il revient (RTI) à ce qu'il faisait avant d'être interrompu.

Le Commodore se sert d'interruptions pour exécuter son système d'exploitation. Le 64 génère 60 interruptions par seconde. Lors de chaque interruption, les rythmeurs du système sont mis à jour, le clavier est analysé, et ainsi de suite. Par conséquent, le système d'exploitation de ces machines possède des horloges et des programmes de traitement associés qui génèrent des interruptions.

Les interruptions du système peuvent servir à exécuter le programme de génération d'impulsions. Les interruptions du Commodore 64 peuvent être interceptées en changeant le vecteur d'interruption. Ce vecteur — une adresse à 2 octets contenue dans deux cellules consécutives — indique au processeur l'emplacement de la routine de gestion d'interruption. En changeant cette adresse afin de pointer la routine d'émission d'impulsions, et en redirigeant le processeur à la fin de l'impulsion vers le programme habituel de traitement d'interruptions du système, le processeur générera une impulsion lors de chaque interruption — c'est-à-dire soixante fois par seconde.

Nous donnons ici les versions BASIC et assembleur d'un programme servant à commander un servomoteur à partir du Commodore 64.

Ils servent également...



La déviation du « volant » attaché à l'arbre du servomoteur est déterminée par la durée de l'impulsion. Si l'impulsion est répétée à des intervalles d'environ 20 ms, le volant sera maintenu en position par le moteur (ce « verrouillage de position » rend le servomoteur idéal pour des applications de commande électro-mécaniques). La plupart des micro-ordinateurs génèrent des interruptions à des intervalles compris entre 10 et 20 ms, ce signal de « rafraîchissement » peut être envoyé sans faire référence au programme de commande BASIC, en associant un code machine au système d'exploitation via le vecteur d'interruption. Autre avantage du servomoteur : une seule ligne de données est requise pour envoyer les impulsions de commande et de rafraîchissement. (Cl. Kevin Jones.)

Commande d'un servomoteur avec le Commodore 64

La première partie du listage, source du Commodore 64 pour la commande d'un servomoteur, illustre comment les vecteurs d'interruption (adresses 788 et 789) sont modifiés. Cela ne peut être fait à l'aide du BASIC, car une interruption pouvant survenir pendant cette modification entraînerait un arrêt anormal du système. Notez que les interruptions sont mises hors fonction (SEI) pendant la modification et sont autorisées de nouveau à l'aide de CLI.
Le reste du code est la routine de gestion d'interruption destinée à la commande d'un seul servomoteur.
Le programme d'appel BASIC renferme tout ce qui est nécessaire pour charger les routines de code machine, pour préparer le port utilisateur, et pour écrire dans l'adresse de mémoire $3000 (12288) des vaieurs correspondant à la touche pressée (1 à 9). Un moteur connecté au port utilisateur doit alors gagner une position proportionnelle à la valeur de la touche. Presser la lettre E met fin au programme.
Si vous avez un assembleur, tapez le listage source et assemblez-le en un fichier objet qui pourra être chargé ultérieurement par le programme d'appel BASIC. Sinon, tapez le programme chargeur BASIC du code machine et exécutez-le pour charger le code en mémoire. Tapez NEW avant de charger et d'exécuter le programme d'appel BASIC. Si vous utilisez le chargeur BASIC, les lignes 30 et 40 peuvent être supprimées.
Remarque : il est extrêmement important de savoir que s'il y a problème dans un programme qui utilise des interruptions, cela peut compromettre le fonctionnement global du système. Vous devrez mettre le système hors tension et le remettre sous tension pour rétablir un fonctionnement normal. Il est donc essentiel de sauvegarder le programme avant de l'exécuter.

Code source

```
;++++++++++++++++++++++++++
;++++++++++++++++++++++++++
;++                      ++
;++    CONDUITE CBM      ++
;++  D'UN SERVOMOTEUR    ++
;++                      ++
;++++++++++++++++++++++++++
;++++++++++++++++++++++++++
;
PORT = 56579 ; USER PORT DATA REG
ANGLE=12288  ; ANGLE VALUE LOCATION
;
*=$0334
;
      SEI         ;INTERRUPTS OFF
      LDA $0314   ;EXXISTING IRQ VECOTOR
      LDX $03C4
      STA $03C4
      STA $0314
      LDA $0315
      LDX $03C5
      STA $03C5
      STX $0315
;
      CLI         ;INTERRUPTS BACK ON
      RTS
;
;++++ EVENT HANDLER ++++
;
      PHP
      PHA
      TYA         ;SAVE REGISTERS
      PHA         ;ON STACK
      TXA
      PHA
      LDA #$FF
      STA PORT
      LDY #$FF
LOOP
      DEY         ;DELAY LOOP
      BNE LOOP    ;APPROX 1MSEC
;
      LDY ANGLE
LOOP1
      DEY         ;COUNT OUT PULSE
      BNE LOOP1
;
      LDA #$00
      STA PORT    ;ZERO DATA REGISTER
;
      PLA
      TAX         ;RESTORE REGISTER
      PLA         ;VALUES
      TAY
      PLA
      PLP
;
      JMP $EA31
```

Programme chargeur basic

```
10 REM **** Chargeur BASIC pour ****
*
20 REM **** un seul servo ****
*
30 :
40 FOR I=820 TO 882
50 READ A:POKE I,A
60 CC=CC+A
70 NEXT I
80 READ CS:IF CC  CS THENPRINT
"CONTROLE DES ERREURS":STOP
100 DATA120,173,20,3,174,196,3,1
41,196
110 DATA3,141,20,3,173,21,3,174,
197,3
120 DATA141,197,3,142,21,3,88,96
,8,72
130 DATA152,72,138,72,169,255,14
1,3
140 DATA221,160,255,136,208,253,
172,0
150 DATA48,136,208,253,169,0,141
,3,221
160 DATA104,170,104,168,104,40,7
6,49
170 DATA234
180 DATA170:REM*CONTROLE*
```

Programme d'appel basic

```
10 REM **** un servomoteur ****
20 :
30 DN=8:REM IF CASSETTE THEN
DN=1
40 IF A=0 THEN A=1:LOAD"SINGSERV
.HEX",8,1
50 POKE 964,79: POKE965,3:REM
POINTER PROGRAMME IRQ
60 DDR=56577: POKE DDR,255:
REM TOUTES SORTIES
70 MC=820:SYS MC: REM SOLLICITER
VECTEUR IRQ
80 POKE 53265,PEEK(53265)AND239:
REM EFFACER ECRAN
90 :
100 GET K$:IF K$="" THEN100:REM
ATTENDRE TOUCHE
110 REM ** MODIFIER POSITION DU MOTEUR
**
120 IF ASC(K$) 48 AND ASC(K$) 58
THEN POKE 12288,VAL(K$)*20
130 IF K$  "E" THEN 80:REM 'E'
POUR EXIT
140 END
```

Sens de l'espace

LOGO est un langage très utile pour étudier la représentation de formes dans l'espace. Pour effectuer des transformations de formes, étudions une procédure qui modifie d'autres procédures.

Il existe quatre sortes de transformations sur une figure à deux dimensions qui laissent la forme inchangée (même si la position change). Ce sont la translation, la rotation, la réflexion et la réflexion-décalage. Le dessin montre comment la position de la figure change dans chaque cas.

On dit d'une figure qu'elle est symétrique lorsque l'on peut la transformer selon une ou plusieurs de ces opérations, sans affecter sa position et sa forme. Les formes finies (comme les polygones ou les lettres de l'alphabet) doivent s'appuyer sur des symétries de réflexion et rotation, car la translation et la réflexion-décalage modifieraient leur position.

Pour étudier et mettre en pratique ces symétries, il est utile d'avoir des procédures LOGO qui réalisent des réflexions et des rotations de formes. Commençons par effectuer une réflexion d'une forme sur une ligne qui passe par l'origine avec une direction déterminée.

Cela sera plus facile si nous supposons que la procédure qui trace la figure est transparente à la forme tracée (c'est-à-dire qu'elle retourne la tortue à sa position et à sa direction de départ). Le travail se décompose en deux parties : trouver les coordonnées et la direction du point de départ de la réflexion, point qui correspond au départ de la forme originale; ensuite, avant de tracer la forme, inverser le sens de tous les ordres de virage dans la procédure de tracer (aller à droite devenant aller à gauche et réciproquement). Une manière de réaliser ce dernier point est de remplacer tous les ordres DROITE, GAUCHE de la procédure, par une procédure, TOURNE, définie de la sorte :

```
POUR TOURNE :A
  DR : DIR * :A
FIN
```

Nous pouvons maintenant définir un carré de la manière suivante :

```
RÉPÈTE 4 [AV 50 TOURNE 90]
```

Pour utiliser cette procédure, il nous faut d'abord affecter à la variable globale DIR la valeur 1. Ainsi FAIS «DIR 1 CARRÉ tracera un carré. Pour effectuer une réflexion du carré selon l'axe des *y*, tapons FAIS «DIR (–1) et ensuite CARRÉ.

La procédure qui positionne la tortue avant le dessin de la réflexion fait appel à la trigonométrie :

```
POUR RÉFLEXION :A
  FAIS «S SENS
  FAIS «XANCIEN XCOOR
  FAIS «ANGLE (ATAN :YANCIEN :XANCIEN) –90 + :A
  FAIS «REMPLACE
  CARRETAN (XANCIEN * :XANCIEN + :YANCIEN * :YANCIEN)
  PL DÉTERMINEXY 0 0
  DÉTERMINE :A + :ANGLE
  AV :R
  DÉTERMINE 2 * :A – :S
  PB
  FAIS «DIR :DIR * (–1)
FIN
```

Cette procédure peut être utilisée pour voir ce que donne la réflexion sur plusieurs lignes qui passent par l'origine. Essayez :

```
FAIS «DIR 1
PL DÉTERMINEXY 0 0
CARRÉ
RÉFLEXION 60
CARRÉ
```

Lorsque la forme réfléchie recouvre complètement la forme originale, on dit qu'il y a « symétrie réfléchie ». Essayez :

```
FAIS «DIR 1
PL DÉTERMINE 0 0 PB
CARRÉ
RÉFLEXION 45
CARRÉ
```

On pourrait aussi écrire une procédure similaire pour effectuer la rotation d'une forme autour d'un point donné et selon un angle déterminé.

Certains motifs, comme ceux des papiers peints, utilisent la même forme géométrique de manière répétitive. Il est possible de créer des effets de translation et de réflexion-décalage qui déplacent l'ensemble du motif sans le modifier. Nous nous contenterons ici des motifs faisant intervenir des translations le long d'une seule ligne.

Les combinaisons des quatre transformations de base aboutissent à sept motifs à partir d'une ligne droite. Ces divers cas forment les motifs de notre exemple. Nous avons écrit les procédures qui dessinent ces sept motifs à partir de n'importe quelle FIGURE, en utilisant les procédures DÉPLACE pour la translation, TOURNE pour la rotation, et REMPLACE.MOTIF qui change tous les virages à DRoite de FIGURE en virages à GAuche, et réciproquement.

Nous avons utilisé les possibilités LOGO de traitement de listes pour la procédure REMPLACE.FIGURE utilisée pour réécrire FIGURE. La procédure utilisée est la suivante :

```
POUR RÉÉCRIS :PROC
   RÉSULTAT PROC.RÉÉCRIS TEXTE :PROC
FIN
```

RÉÉCRIS prend le texte d'une procédure donnée et le restitue sous un autre nom. Elle suppose que la procédure en question est écrite en primitives LOGO et ne contient pas de sous-procédures. RÉÉCRIS comprend un appel aux procédures suivantes :

```
POUR PROC.RÉÉCRIS :TEXTE
   SI :TEXTE = [] ALORS RÉSULTAT []
   RÉSULTAT METSF RÉÉCRIS.LIGNE PREMIER :TEXTE
   RÉÉCRIS PROC. SAUFPREMIER :TEXTE
FIN
```

Cette procédure divise le texte en ses lignes constitutives en appelant la procédure suivante :

```
POUR RÉÉCRIS.LIGNE :LIGNE
   SI :LIGNE = [] ALORS RÉSULTAT []
   SI LISTE? PREMIER :LIGNE ALORS RÉSULTAT METSF
   RÉÉCRIS.LIGNE PREMIER :LIGNE RÉÉCRIS.LIGNE
   SAUFPREMIER :LIGNE
   RÉSULTAT METSF
   MODIFIE.MOT PREMIER :LIGNE
   RÉÉCRIS.LIGNE SAUFPREMIER :LIGNE
FIN
```

RÉÉCRIS.LIGNE traite le texte ligne par ligne, transmettant chaque mot à MODIFIE.MOT qui effectue le travail. La ligne qui commence par SI LISTE? est nécessaire au cas où MOTIF contiendrait une instruction RÉPÈTE. Si vous excluez cette éventualité de la procédure MOTIF, vous pouvez retirer cette ligne. Le listage de MODIFIE.MOT est le suivant :

```
POUR MODIFIE.MOT :MOT
   SI (AUMOINSUN :MOT = «DR :MOT = «DROITE) ALORS
   RÉSULTAT «GAUCHE
   SI (AUMOINSUN :MOT = «GA :MOT = «GAUCHE) ALORS
   RÉSULTAT «DROITE
   RÉSULTAT :MOT
FIN
```

Cette procédure vérifie tous les mots et effectue les modifications nécessaires. Ayant saisi toutes ces procédures, voyons comment elles fonctionnent. Il nous faut d'abord définir une forme simple, par exemple :

```
POUR TRI
   RÉPÈTE 3 [AV 50 DR 120]
FIN
```

Frappez alors DÉFINIS «REDEF RÉÉCRIS «TRI, et appelez REDÉF. Sa définition sera :

```
POUR REDÉF
   RÉPÈTE 3 [AV 50 GA 120]
FIN
```

Il est tout à fait possible d'écrire une procédure RÉÉCRIS plus générale qui réécrive également toutes les sous-procédures appelées par la principale. Si vous essayez, prenez garde aux procédures récursives ! Il vous faudra aussi tester chaque mot pour savoir s'il s'agit d'un nom de procédure.

Les sept motifs à bâtonnets

Il serait possible (et mathématiquement élégant), d'établir les motifs découlant des procédures et obtenus par les quatre transformations de base. Les procédures de dessin des motifs utilisent trois sous-procédures très utiles :

```
POUR POSITIONNE
   CACHETORTUE
   LÈVEPLUME
   DÉTERMINEXY =125
   POSEPLUME
FIN
```

Cela positionne la tortue à l'extrémité gauche de l'écran, prête à dessiner.

```
POUR DÉPLACE
   LÈVEPLUME
   DR 90
   AV 50
   GA 90
   POSEPLUME
FIN
```

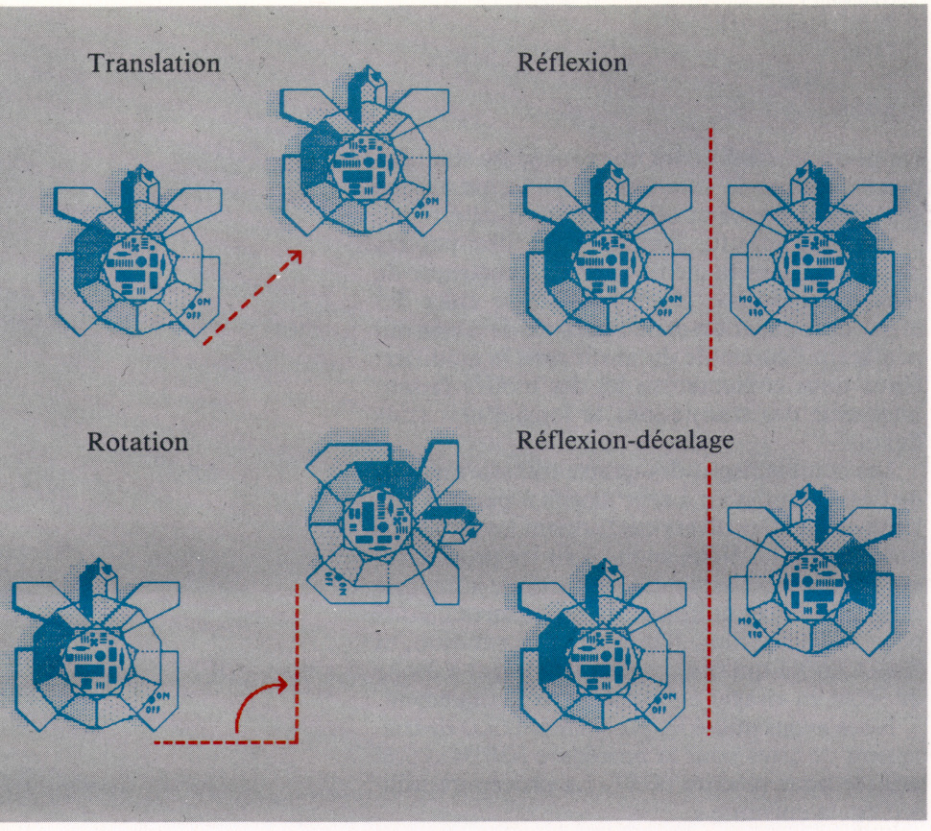
DÉPLACE effectue la translation désirée.

```
POUR TOURNE
  DR 180
FIN
```

TOURNE effectue la rotation voulue.

Pour utiliser ces procédures, commencez par définir une procédure de tracé de formes, FORME par exemple. Vous pouvez alors tracer le premier motif géométrique à l'aide de FORME en le considérant comme la figure voulue, en tapant :

```
MOTIF1 «FIGURE
```



La comète de l'isométrie
Les transformations qui modifient la position d'un objet, mais non sa forme, sont connues sous le terme d'isométries. Les quatre transformations isométriques sont : la translation, la rotation, la réflexion et la réflexion-décalage.
La translation est un simple déplacement de la figure d'origine.
La rotation fait tourner la figure autour d'un point.
La réflexion suppose la transposition de la figure au-delà d'une ligne miroir, de sorte que chaque point de la figure finale soit à la même distance de la ligne que ceux de la figure d'origine. La réflexion-décalage est une combinaison de la réflexion et du décalage.
Si la translation et la rotation ménagent la disposition de la figure, la réflexion et la réflexion-décalage la modifie : pour le comprendre, essayez de faire se réfléchir une lettre dans un miroir par exemple.
(Cl. Ian McKinnell.)

Sept motifs à bâtonnets

Pour exécuter les procédures des motifs, vous avez besoin des procédures suivantes : RÉÉCRIS, PROC, RÉÉCRIS.LIGNE, MODIFIE.MOT, POSITIONNE, DÉPLACE et TOURNE. Les sept motifs possibles sont les suivants :

```
POUR MOTIF1 :PROC
   DÉFINIS «FIGURE TEXTE :PROC
   POSITIONNE
   RÉPÈTE 6 [FIGURE DÉPLACE]
FIN
POUR MOTIF2 :PROC
   DÉFINIS «FIGURE TEXTE :PROC
   DÉFINIS «REMPLACE.FIGURE RÉÉCRIS.PROC TEXTE
   POSITIONNE
   RÉPÈTE 3 [FIGURE DÉPLACE TOURNE
   REMPLACE.FIGURE
   TOURNE DÉPLACE]
FIN
POUR MOTIF3 :PROC
   DÉFINIS «FIGURE TEXTE :PROC
   DÉFINIS «REMPLACE.FIGURE RÉÉCRIS.PROC
   TEXTE :PROC
   POSITIONNE
   RÉPÈTE 6 [FIGURE REMPLACE.FIGURE DÉPLACE]
FIN
POUR MOTIF4 :PROC
   DÉFINIS «FIGURE TEXTE :PROC DÉFINIS
   «REMPLACE.FIGURE RÉÉCRIS.PROC TEXTE :PROC
   POSITIONNE
   RÉPÈTE 6 [FIGURE TOURNE FIGURE TOURNE DÉPLACE]
FIN
POUR MOTIF5 :PROC
   DÉFINIS «FIGURE TEXTE :PROC DÉFINIS
   «REMPLACE.FIGURE RÉÉCRIS.PROC TEXTE :PROC
   POSITIONNE
   RÉPÈTE 3 [FIGURE REMPLACE.FIGURE DÉPLACE
   TOURNE FIGURE REMPLACE.FIGURE TOURNE
   DÉPLACE]
FIN
POUR MOTIF6 :PROC
   DÉFINIS «FIGURE TEXTE :PROC DÉFINIS
   «REMPLACE.FIGURE RÉÉCRIS.PROC TEXTE :PROC
   POSITIONNE
   RÉPÈTE 6 [FIGURE TOURNE REMPLACE.FIGURE
   TOURNE DÉPLACE]
FIN
POUR FIGURE7 :PROC
   DÉFINIS «FIGURE TEXTE :PROC DÉFINIS
   «REMPLACE.FIGURE RÉÉCRIS.PROC TEXTE :PROC
   POSITIONNE
   RÉPÈTE 6 [FIGURE REMPLACE.FIGURE TOURNE FIGURE
   REMPLACE.FIGURE TOURNE DÉPLACE]
FIN
```

Ces procédures doivent être exécutées à l'aide d'une procédure de dessin appropriée. La figure graphique que nous utilisons est la suivante :

```
POUR BÂTONNET
   AV 50
   DR 90
   AV 20
   AR 20
   GA 90
   AR 50
FIN
```

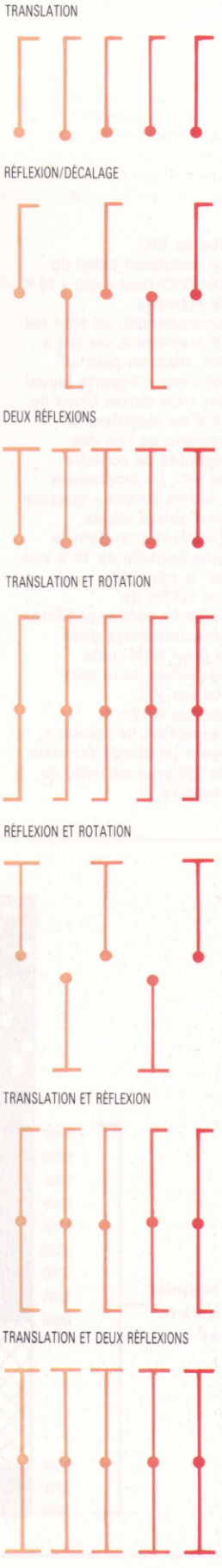
Ou comme alternative :

```
POUR FIG
   DR 30
   AV 20
   GA 50
   AV 20
   DR 90
   AV 10
   RÉPÈTE 4[AV 20 DR 90]
   AR 10
   GA 90
   AR 20
   DR 50
   AR 20
   GA 30
FIN
```

Variantes logo

ANTAN et VERS n'existent pas dans LOGO Atari, et ils n'ont pas d'équivalents simples. Cela affecte les procédures RÉFLEXION et ROTATION, mais non les procédures MOTIF.
LOGO Atari ne comporte pas les primitives TEXTE et DÉFINIS, même si le manuel Atari donne un moyen de les définir. Vous pourriez peut être écrire FIGURE.DR en modifiant simplement FIGURE à l 'aide de l'éditeur.

Logomotif



Sept du même genre Les quatre transformations isométriques de base peuvent se combiner de diverses manières. On aboutit aux sept motifs distincts que l'on voit ici. Pour chacun d'eux, nous commençons par montrer la figure normale en bâtonnet, toutes les translations étant faites selon l'axe des *x*. (Cl. Liz Dixon.)

Échec et mat

Avec le Commodore 64, on peut translater facilement l'emplacement des données d'écran en mémoire. Voyons une routine pour concevoir et stocker plusieurs (jusqu'à huit) affichages écran.

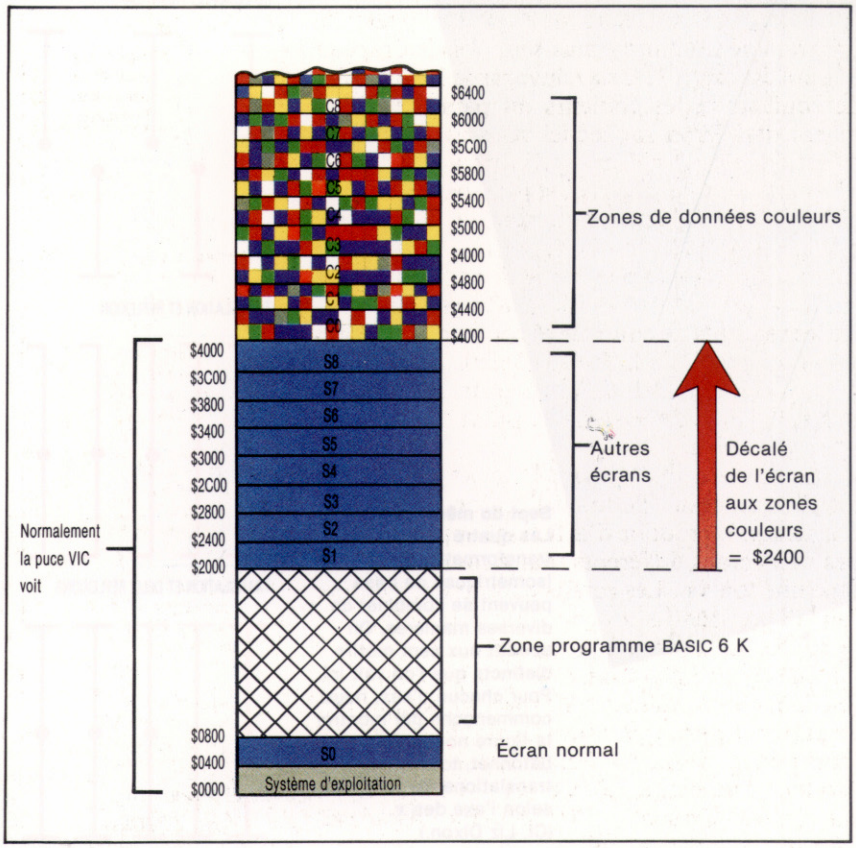
Vue de VIC
Le contrôleur vidéo du C64 (VIC) peut « voir » 16 K de mémoire. Normalement, ce sont les 16 premiers K, de $0000 à $3FFF, mais on peut lui faire voir n'importe lequel des trois autres blocs de 16 K en modifiant le contenu de l'un des registres de contrôle de VIC. Le programme « autres écrans » instaure neuf autres tables d'implantation dans la zone normale de 16 K vue par la puce VIC. Les tables de couleurs correspondantes sont contenues dans la zone RAM juste au-dessus de la zone vue par VIC; chaque écran, à l'exception de l'écran 0, ayant un décalé constant de $2400 pour sa table de couleurs.

L'affichage écran et le maniement de lutins sont contrôlés par une puce spéciale du Commodore 64, appelée VIC II. La puce VIC a accès à différentes parties de mémoire pour obtenir l'information à partir de laquelle elle crée l'affichage. Ces zones incluent la ROM caractère, où sont obtenues les formes de caractères; la RAM couleur, comportant l'information de couleur; et la RAM écran. Cette dernière contient les informations concernant les caractères à afficher dans n'importe lequel des mille emplacements (25 lignes par 40 colonnes) qui forment l'écran.

Lorsque le C64 est allumé, la puce VIC admet que l'écran est placé dans les 1000 octets commençant à l'emplacement 1024 ($0400), et accède à cette zone pour obtenir son information écran initiale. Cependant, en modifiant la valeur d'un registre, nous pouvons réorienter la puce vers une autre zone de mémoire — normalement les 16 premiers kilo-octets en mémoire. Les quatre bits supérieurs, à l'emplacement 53272 ($0018) du registre contrôle VIC, déterminent lequel des blocs d'un kilo-octet, sur les 16 de la zone, est interprété comme écran. Le tableau suivant montre les valeurs de bits correspondant à chacune des 16 positions d'écran possibles :

Configuration de bits	Début d'écran	
0000XXXX	0	$0000
0001XXXX	1024	$0400*
0010XXXX	2048	$0800
0011XXXX	3072	$0C00
0100XXXX	4096	$1000
0101XXXX	5120	$1400
0110XXXX	6144	$1800
0111XXXX	7168	$1C00
1000XXXX	8192	$2000
1001XXXX	9216	$2400
1010XXXX	10240	$2800
1011XXXX	11264	$2C00
1100XXXX	12288	$3000
1101XXXX	13312	$3400
1110XXXX	14336	$3800
1111XXXX	15360	$3C00

* = Position par défaut



Kevin Jones

Pour permettre à la puce VIC de déplacer l'écran dans une autre zone, il faut changer les quatre bits supérieurs de l'emplacement 53272 ($D018) en les valeurs indiquées dans le tableau. Cependant, il ne faut pas modifier les quatre bits inférieurs (la partie XXXX de la configuration de bits dans le tableau), car ils contrôlent une autre fonction. Pour annuler les autres bits supérieurs sans changer la valeur des quatre inférieurs, nous devons appliquer la fonction ET au contenu du registre avec 15 (00001111 en binaire). Après cela, on peut faire OU sur le nouveau contenu avec la valeur voulue. Pour positionner l'écran dans la dernière zone que peut voir la puce VIC — c'est-à-dire à partir de 15360 ($3C00) — il faut faire OU sur le contenu du registre avec 240 (11110000 en binaire). En BASIC, l'instruction POKE sera ainsi utilisée :

```
POKE 53272,(PEEK(53272)AND15)OR240
```

Avant de pouvoir écrire quoi que ce soit sur notre nouvel écran, il faudra aussi dire au système d'exploitation que la position de l'écran a changé. Pour cela, on place l'octet hi de l'adresse de départ du nouveau début d'écran en 648 ($0288). Pour l'écran le plus élevé, c'est $3C, et l'on y parvient en BASIC en divisant l'adresse de début d'écran par 256.

Nous pouvons utiliser l'aptitude du C64 à déplacer son écran pour produire différentes possibilités intéressantes. En particulier, on peut changer rapidement l'affichage. Le problème est que, en déplaçant l'écran, il faut aussi déplacer la RAM couleur correspondante, parce que l'affichage n'aura d'effet que si cette RAM contient les données correspondant à l'écran affiché. Nous pouvons mettre en place plusieurs zones écran différentes en mémoire et les échanger rapidement, mais il n'y a qu'une seule zone de RAM couleur inamovible; aussi, pour garder plusieurs écrans, nous devons réserver des zones de mémoire pour conserver les 1000 octets de données couleurs de chaque écran. Si nous voulons afficher un écran, nous devons copier cette information dans la RAM couleur, et sauvegarder cette donnée dans l'une des zones de mémoire que nous avons désignées, pour y mettre la RAM couleur, avant de passer à un écran différent.

La tâche principale de cette faculté est d'organiser la mémoire pour différents écrans (avec leurs zones de données couleurs correspondantes), et d'effectuer le transfert de blocs de mémoire. Comme la puce VIC peut « voir » 16 K de mémoire, nous pouvons concevoir un système qui nous autorise à avoir jusqu'à huit écrans différents et un programme BASIC assez important. Le diagramme montre la disposition de mémoire utilisée à cet effet.

Pour s'assurer qu'aucun des écrans ou zones de couleur ne sont écrasés par un programme, il faut abaisser le haut de la mémoire. L'instruction suivante dans notre programme d'appel fera cela :

```
POKE 55,0: POKE 56,32:CLR
```

L'adresse de base de tout écran peut être calculée, à partir de son numéro, par cette formule :

```
Base Écran = $1C00 + ($0400 * Numéro d'écran)
```

L'adresse de base de la zone couleur correspondante peut être calculée simplement en additionnant un décalé de l'adresse de base de l'écran. Voici la formule :

```
Base Couleur = $2400 + Base Écran
```

Les zones couleurs peuvent être placées n'importe où en RAM, mais il est plus commode de les positionner juste au-dessus du dernier écran qui peut être vu par la puce VIC. On remarquera qu'une zone couleur pour l'écran 0 — écran normal — est incluse. Pour cette zone couleur particulière, le décalé sera différent, et notre programme devra en tenir compte.

Le registre de contrôle de VIC et celui du système d'exploitation peuvent être mis pour n'importe quel écran par :

```
Registre VIC : $70 + ($10 * Numéro d'écran)
Registre SE = Octet-hi d'adresse de base d'écran
```

En outre, pour manier l'emplacement des registres et effectuer les transferts appropriés de données couleurs vers et à partir de la zone RAM couleur, le programme stocke aussi la couleur du fond et de la marge de l'écran. Ces deux caractéristiques sont contrôlées par une paire de registres dans la puce VIC : 53280 ($D020) contrôle la couleur de la marge et 53281 ($D021) contrôle le fond d'écran, ou couleur du papier.

Modes opératoires

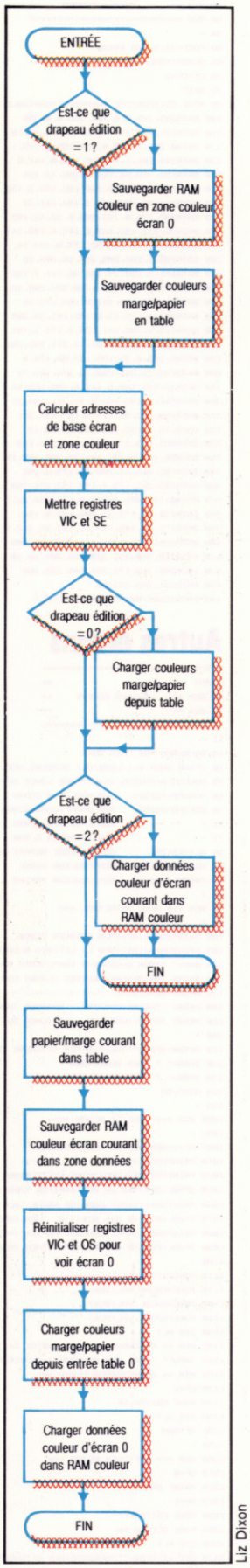
Le programme distingue deux modes opératoires : on peut soit éditer, soit afficher un écran choisi. Dans les deux cas, le numéro d'écran à utiliser doit être POKé en 49152 ($C000). Afin que le même appel SYS puisse être fait, nous utilisons un drapeau spécial pour indiquer mode choisi. Ce drapeau est mis en POKant vers l'emplacement 49153 ($C001).

0 – indique le mode affichage
1 – indique le mode éditeur

Le mode éditeur fonctionne de manière inhabituelle, pour permettre toutes les fonctions de l'éditeur d'écran (comme changer les couleurs de texte, se mettre en mode inverse et effacer l'écran). Le programme doit d'abord être appelé et le drapeau édition mis à 1. Cela sauvegarde alors la zone couleur normale de l'écran, ainsi que les couleurs de papier et de marge, règle convenablement les régistres de la puce VIC et du système d'exploitation, puis retourne au BASIC. A ce stade, le programme d'appel BASIC reprend la main, réinitialise le curseur, puis utilise l'instruction INPUT en BASIC. Cette instruction attend un caractère de Retour-Chariot (ASCII 13) avant de continuer. Entre-temps, toutes les fonctions de l'éditeur d'écran peuvent être utilisées normalement pour éditer l'écran choisi, en appuyant sur Retour-Chariot après chaque tâche.

Le programme BASIC doit alors rappeler le programme utilitaire; mais cette fois le drapeau édition est mis à 3. Cela sauvegarde les données de couleurs et les couleurs du papier et de la marge de l'écran sur lequel vous avez travaillé avant de restaurer l'écran. Si l'on veut changer la couleur soit de la marge soit du papier, il faut alors POKEr respectivement dans les emplacements 49154 ($C002) et 49155 ($C003). Bien que la routine soit conçue pour cette tâche particulière, elle incorpore des routines générales pour mettre les registres de contrôle et copier des données vers ou à partir de la RAM couleur. Il ne serait donc pas difficile de faire un programme pour nos propres spécifications à partir de ces routines générales.

Le programme d'appel BASIC est conçu pour afficher un menu donnant l'option d'éditer ou d'afficher. La routine d'affichage appelle chacun des huit écrans différents tour à tour, lorsqu'on tape une touche. Les écrans continuent à défiler jusqu'à ce que la touche Retour-Chariot soit appuyée. Le programme restaure alors l'écran normal et revient au menu. Si l'option édition est sélectionnée, l'utilisateur peut régler les couleurs de papier et de marge pour l'écran choisi, et peut alors utiliser l'éditeur d'écran normalement pour produire une image. Lorsqu'on a fini, on appuie sur Retour-Chariot : l'image sera stockée et l'écran normal restauré.



Liz Dixon

Chargeur basic

```
10 REM ******************************
20 REM **  CHARGEUR BASIC POUR  **
30 REM **      AUTRES ECRANS    **
40 REM ******************************
50 :
60 FORI=49152 TO 49439
65 READA:POKE I,A
70 CC=CC+A
75 NEXT
80 READ CS:IFCS<>CC THENPRINT"CONTROLE ERREUR":STOP
100 DATA255,255,0,0,191,255,0,0,255
110 DATA191,0,0,191,191,0,0,255,191,0
120 DATA0,255,191,0,0,255,255,0,173,1
130 DATA192,201,1,208,30,169,0,141,6
140 DATA192,169,64,141,7,192,32,224
150 DATA192,162,0,32,246,192,173,2,192
160 DATA141,32,208,173,3,192,141,33
170 DATA208,173,0,192,208,6,32,32,193
180 DATA76,139,192,169,0,141,4,192,141
190 DATA5,192,174,0,192,173,5,192,24
200 DATA105,4,202,208,250,24,105,28
210 DATA141,5,192,24,105,36,141,7,192
220 DATA173,0,192,162,4,10,202,208,252
230 DATA24,105,112,141,8,192,173,24
240 DATA208,41,15,13,8,192,141,24,208
250 DATA173,5,192,141,136,2,173,1,192
260 DATA208,6,174,0,192,32,211,192,201
270 DATA2,240,4,32,189,192,96,174,0
280 DATA192,32,246,192,32,224,192,32
290 DATA32,193,169,0,141,6,192,169,64
300 DATA141,7,192,162,0,32,211,192,32
310 DATA189,192,96,173,6,192,133,251
320 DATA173,7,192,133,252,169,0,133
330 DATA253,169,216,133,254,32,3,193
340 DATA96,189,9,192,141,32,208,189,18
350 DATA192,141,33,208,96,173,6,192
360 DATA133,253,173,7,192,133,254,169
370 DATA0,133,251,169,216,133,252,32,3
380 DATA193,96,173,32,208,157,9,192
390 DATA173,33,208,157,18,192,96,162,3
400 DATA160,0,177,251,145,253,136,208
410 DATA249,230,252,230,254,202,48,10
420 DATA208,240,177,251,145,253,160
430 DATA231,208,232,96
440 DATA36606:REM*CONTROLE TOTAL*
```

Autres écrans

```
3 REM  ************************
5 REM  **                    **
10 REM **    AUTRES ECRANS   **
11 REM **                    **
12 REM ************************
13 :
15 DN=8:REM FOR CASS DN=1
20 IFA=0 THEN A=1:LOAD"ALT SCREENS.HEX",DN,1
25 POKE55,0:POKE56,32:CLR:REM LOWER MEMTOP
40 SCNUMB=49152:   REM NUMERO D'ECRAN
70 EDITFG=49153:   REM 0=AFFICHER ECRAN
75 :               REM 1=EDITER ECRAN
77 :               REM 2=COPIER C.RAM
80 ALT=49179:      REM ADRESSE DEPART LANGAGE MACHINE
85 BRDCOL=49154:   REM COULEUR MARGE
87 PAPCOL=49155:   REM COULEUR PAPIER
90 :
95 REM *** MENU PRINCIPAL ***
96 :
100 PRINTCHR$(147):REM EFFACER ECRAN
105 PRINTCHR$(154):REM LT LETTRES BLEUES
110 DN$=" ":REM Q=CURSEUR CARACTERES BAS
130 PRINT TAB(8):DN$;"AUTRES ECRANS C8M 64"
135 PRINTTAB(8):"------------------------"
140 PRINT  TAB(5):DN$:"F1 - EDITEUR IMAGE"
160 PRINT TAB(5):DN$;"F3 - AFFICHER SUITE D'IMAGES"
200 :
210 GETA$:IFA$=""THEN210:REM ATTENDRE TOUCHE
220 IFA$=" " THEN GOSUB1000
230 IFA$=" " THEN GOSUB 1500
260 GOTO100
999 :
1000 REM **** EDITER ECRAN ****
1002 :
1005 EF=1:REM METTRE MODE EDITEUR
1010 PRINTCHR$(147)
1020 PRINTTAB(16):DN$;"MODE EDITEUR:SN$
1050 IFASC(SN$)<48 OR ASC(SN$)>56 THEN 1030
1060 PRINTDN$;:INPUT"COULEUR MARGE":BC$
1070 IFVAL(BC$)=0 AND BC$<>"0" THEN 1060
1080 PRINTDN$;:INPUT"COULEUR PAPIER":PC$
1090 IFVAL(PC$)=0 AND PC$<>"0" THEN 1080
1100 :
1110 POKEEDITFG,EF
1120 POKESCNUMB,VAL(SN$)
1130 POKEBRDCOL,VAL(BC$)
1140 POKEPAPCOL,VAL(PC$)
1150 SYS ALT
1155 REM ** ATTENDRE RETOUR CHARIOT **
1162 INPUT" ";X$: REM S=CURSEUR REINITIALISE
1165 REM ** SAUVEGARDER ECRAN **
1170 EF=2
1175 POKE EDITFG,EF
1180 SYS ALT
1185 RETURN
1190 :
1500 REM *** AFFICHER ECRAN ***
1510 EF=0
1520 PRINT CHR$(147)
1545 SN=1
1550 POKE EDITFG,EF
1555 POKE SCNUMB,SN
1560 SYS ALT
1570 GET X$:IFX$=""THEN1570:REM ATTENDRE TOUCHE
1572 IFX$=CHR$(13)THEN 1580:REM ECRAN NORMAL
1575 SN=SN+1:IFSN<9 THEN 1555
1577 SN=1:GOTO1555
1580 POKESCNUMB,0:SYS ALT
1600 RETURN
```

Commodore 64

```
;++++++++++++++++++
;++++++++++++++++++
;++              ++
;++    AUTRES    ++
;++    ECRANS    ++
;++     POUR     ++
;++    C8M 64    ++
;++              ++
;++++++++++++++++++
;++++++++++++++++++
;
FROM    =$FB            ;0 PAGE
TO      =$FD            ;POINTERS
;
CRAMLO=$00              ;START OF COLOUR RAM
CRAMHI=$08
NCOLLO=$00              ;NORMAL COLOUR
NCOLHI=$40              ;BASE ADDRESS
NSCRHI=$04              ;NORMAL SCREEN BASE ADDRESS
NVCPOK=$10              ;NORMAL VIC REG VALUE
;
BLOCKS=$03              ;NO OF 256 BYTE BLOCKS
EXTRA =$E7              ;EXTRA AMOUNT TO 1000 BYTES
;
COLOFF=$24              ;COLOUR OFFSET HIBYTE
SCROFF=$1C              ;SCREEN OFFSET HIBYTE
VICOFF=$70              ;VIC CTRL REG OFFSET
VCMASK=$0F              ;VIC CTRL REG LOBYTE MASK
VICREG=$0018            ;SCREEN LOCATION CONTROL REG
EDREG=$0288             ;SCREEN EDITOR KERNAL REG
BORDER=$D020            ;BORDER COLOUR REG
PAPER =$D021            ;BACKGROUND COLOUR REG
;
*=$C000                 ;SET LOAD POINTER
;
SCNUMB *=*+1            ;SCREEN NUMBER
EDITFG *=*+1            ;EDIT MODE FLAG
BRDCOL *=*+1            ;BORDER COLOUR
PAPCOL *=*+1            ;PAPER COLOUR
;
SCBASE *=*+2            ;SCREEN BASE STORAGE
CLBASE *=*+2            ;COLOUR BASE STORAGE
VPOKE  *=*+1            ;TEMP STORE FOR VIC NUMBER
BRDTAB *=*+9            ;BORDER COLOURS TABLE
PAPTAB *=*+9            ;PAPER COLOURS TABLE
;
;++++ SAVE SCREEN 0 ++++
        LDA EDITFG
        CMP #$01        ;IF 0 OR 2
        BNE CALC        ;THEN DON'T SAVE
;
        LDA #NCOLLO
        STA CLBASE
        LDA #NCOLHI
        STA CLBASE+1
        JSR SAVE        ;SAVE CRAM TO C0
        LDX #$00
        JSR SAVEBP      ;SAVE B/P REGS
;
        LDA BRDCOL
        STA BORDER      ;SET NEW B/P
        LDA PAPCOL      ;COLOURS
        STA PAPER
;
;++++ CALCULATE COLOUR BASE ++++
;
CALC
        LDA SCNUMB
        BNE NOZERO
        JSR RESET       ;SET NORMAL REGS
        JMP TESTFG
;
NOZERO
        LDA #$00
        STA SCBASE
        STA SCBASE+1    ;INIT SCBASE
;
        LDX SCNUMB      ;LOAD SCREEN NUMBER
        LDA SCBASE+1    ;HI BYTE ONLY
MULT
        CLC
        ADC #$04
        DEX
        BNE MULT
;
        CLC
        ADC #SCROFF     ;ADD SCREEN OFFSET
        STA SCBASE+1
;
        CLC
        ADC #COLOFF     ;ADD COLOUR OFFSET
        STA CLBASE+1
;
;++++ SET VIC AND EDITOR REGISTERS ++++
        LDA SCNUMB
        LDX #$04
MORE
        ASL A
        DEX
        BNE MORE        ;MULT BY 16
;
        CLC
        ADC #VICOFF     ;ADD OFFSET
        STA VPOKE
        LDA VICREG
        AND #VCMASK
        ORA VPOKE
        STA VICREG      ;SET VIC REG
;
        LDA SCBASE+1
        STA EDREG       ;SET EDITOR REG
;
;++++ TEST STATUS OF EDIT FLAG ++++
TESTFG
        LDA EDITFG
        BNE NOLOAD      ;IF 0 DISPLAY MODE
        LDX SCNUMB
        JSR LOADBP      ;LOAD B/P TO REGS
;
;
NOLOAD
        CMP #$02
        BEQ CONT        ;IF 2 THEN SAVE RAM
;
        JSR LOAD        ;LOAD COL TO CRAM
        RTS
;
CONT
        LDX SCNUMB
        JSR SAVEBP      ;SAVE BP REGS
        JSR SAVE        ;SAVE CRAM TO COL
        JSR RESET       ;SET NORMAL REGS
        LDA #NCOLLO
        STA CLBASE      ;LOAD CBASE WITH C0 BASE
        LDA #NCOLHI
        STA CLBASE+1
        LDX #$00
        JSR LOADBP      ;RELOAD ORIG BORD/PAP COLS
        JSR LOAD        ;LOAD NRML SCR COLS
        RTS
;
;+++++ TRANSFER TO RAM S/R ++++
LOAD
        LDA CLBASE
        STA FROM        ;LOAD 0 PAGE PTRS
        LDA CLBASE+1
        STA FROM+1
;
        LDA #CRAMLO
        STA TO
        LDA #CRAMHI
        STA TO+1
;
        JSR COPY        ;COPY RAM AREA
        RTS
;
;++++ LOAD BORDER & PAPER COLS S/R ++++
LOADBP
        LDA BRDTAB,X
        STA BORDER
        LDA PAPTAB,X
        STA PAPER
        RTS
;
;++++ TRANSFER FROM RAM S/R ++++
SAVE
        LDA CLBASE
        STA TO          ;LOAD 0 PAGE PTRS
        LDA CLBASE+1
        STA TO+1
;
        LDA #CRAMLO
        STA FROM
        LDA #CRAMHI
        STA FROM+1
;
        JSR COPY        ;COPY RAM AREA
        RTS
;
;++++ SAVE BORDER & PAPER COLS S/R ++++
SAVEBP
        LDA BORDER
        STA BRDTAB,X
        LDA PAPER
        STA PAPTAB,X
        RTS
;
;++++ COPY 1000 BYTES S/R ++++
;
COPY
        LDX #BLOCKS
        LDY #$00
NEXT
        LDA (FROM),Y
        STA (TO),Y
        DEY
        BNE NEXT
NXBLOC INC FROM+1
        INC TO+1
        DEX
        BMI FINISH
        BNE NEXT
        LDA (FROM),Y
        STA (TO),Y
        LDY #EXTRA      ;EXTRA BYTES
        BNE NEXT
FINISH
        RTS
;
;++++ RESET VIC AND EDIT REGS S/R ++++
RESET
        LDA #NCOLLO
        STA CLBASE
        LDA #NCOLHI
        STA CLBASE+1
        LDA VICREG
        AND #VCMASK
        ORA #NVCPOK
        STA VICREG      ;RESTORE VIC REG
        LDA #NSCRHI
        STA EDREG       ;RESTORE EDREG
        RTS
```

Page manquante
(publicité)

Page manquante
(publicité)